MPI: Non-blocking point-to-point communication

CSCS-USI Summer School 2014

Themis Athanassiadou

Serpiano, Switzerland

Imagine the following situation…

- **Each process sends a message to the right neighbor, and receives from the left… (live demo!)**
- **The code for each process is: MPI_Send(…., to_right,…)**
  MPI_Recv(…., from_left…)

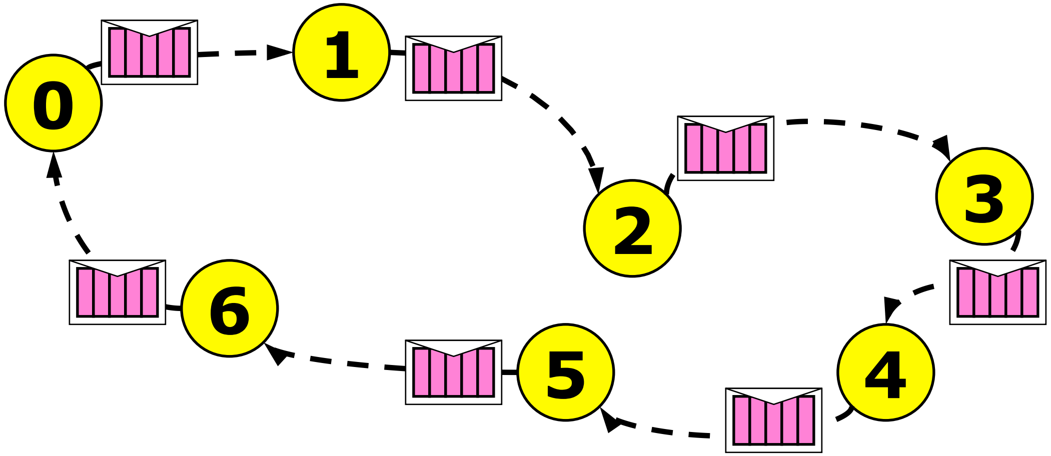


Image: Rolf Rabenseifner, HLRS

Imagine the following situation…

- **Each process sends a message to the right neighbor, and receives from the left… (live demo!)**

- **The code for each process is: MPI_Send(…., to_right,…)**
  MPI_Recv(…., from_left…)

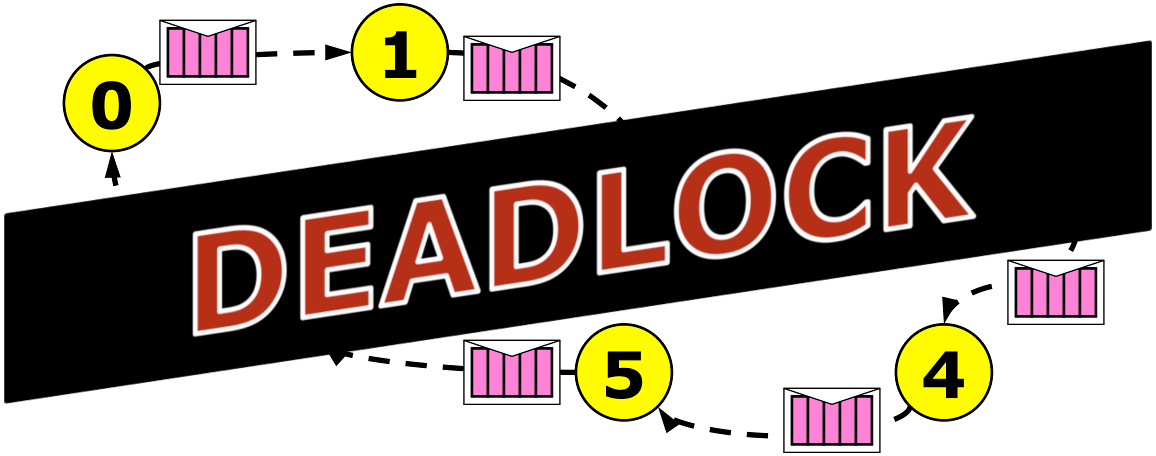


Image: Rolf Rabenseifner, HLRS

Non-blocking communication to the rescue…

Non-blocking mode (MPI_Isend, MPI_Irecv):

– Non-blocking send and receive routines behave similarly - they will return Immediately. They do not wait for any communication events to complete

– Non-blocking operations simply "request" the MPI library to perform the operation when it can. The user cannot predict when this will happen.

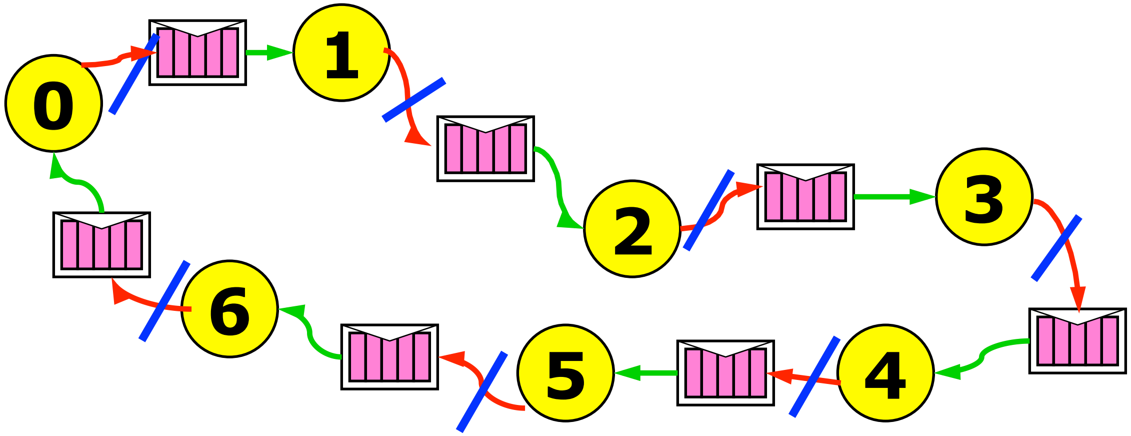
– It is unsafe to modify the application buffer until communication is actually completed. Completion is ensured by the wait functions.

– Non-blocking communications are primarily used to overlap computation with communication.

The ring example: Avoiding the deadlock

The code now for each process is:

MPI_ISend(…., to_right,…) →
MPI_Recv(…., from_left…) →
MPI_Wait (…….) /



MPI_Isend/ MPI_Irecv (FORTRAN version)

```
call MPI_ISEND(buf, count, type, dest, tag, comm, req, ierr)
call MPI_IRECV(buf, count, type, source, tag, comm, req, ierr)
```

- buf : array
- count (INTEGER) number of element of buf to be sent
- type MPI type of buf (MPI_INTEGER, MPI_REAL etc)
- dest (INTEGER) rank of the destination process
- tag (INTEGER) number identifying the message
- comm (INTEGER) communicator of the sender and receiver
- req (INTEGER) **output**, identifier of the communications handle
- ierr (INTEGER) **output**, error code (if ierr=0 no error occurs)

MPI_Isend, MPI_Irecv (C Version)

```
int MPI_Isend(void *buf, int count, MPI_Datatype
      type, int dest, int tag, MPI_Comm comm, MPI_Request *req);
```

```
int MPI_Irecv (void *buf, int count, MPI_Datatype
               type, int source, int tag, MPI_Comm comm, MPI_Request  *req);
```

Waiting for completion

- **Fortran:**
  MPI_WAIT(req, status, ierr)
  MPI_WAITALL (count,array_of_requests,array_of_statuses, ierr)

- A call to this subroutine causes the code to wait until the communication referred to by req is complete.
- req(INTEGER):input/output, identifier associated to a communications event(initiated by MPI_ISEND or MPI_IRECV).
- Status(INTEGER) array of size MPI_STATUS_SIZE, if req was associated to a call to MPI_IRECV, status contains information on the received message, otherwise status could contain an error code.
- ierr(INTEGER) output, error code (if ierr=0 no error occurs).

- **C:**
  int MPI_Wait(MPI_Request *req, MPI_Status *status)
  int MPI_Waitall (count,&array_of_requests,&array_of_statuses)

Thank you for your attention.