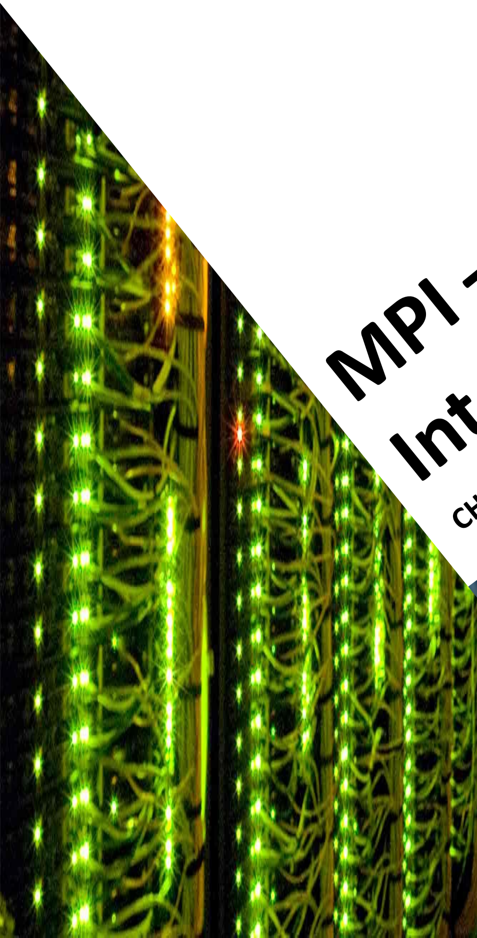


# MPI – Message Passing Interface

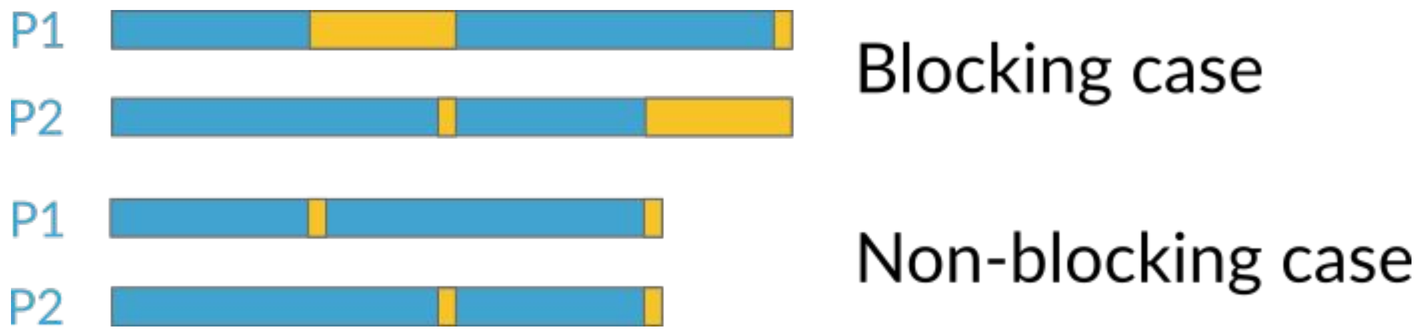
CHAPTER 4

# Outline

1. MPI Overview
2. Basic Structure of a MPI program
3. Messages and Point-to-Point Communication
4. **Non-blocking Communication**
5. Derived Data Types
6. Collective Communication

# Non-blocking Communication

## Why nonblocking communication?



**Blocking Communication** requires processes to stop working to send and receive messages

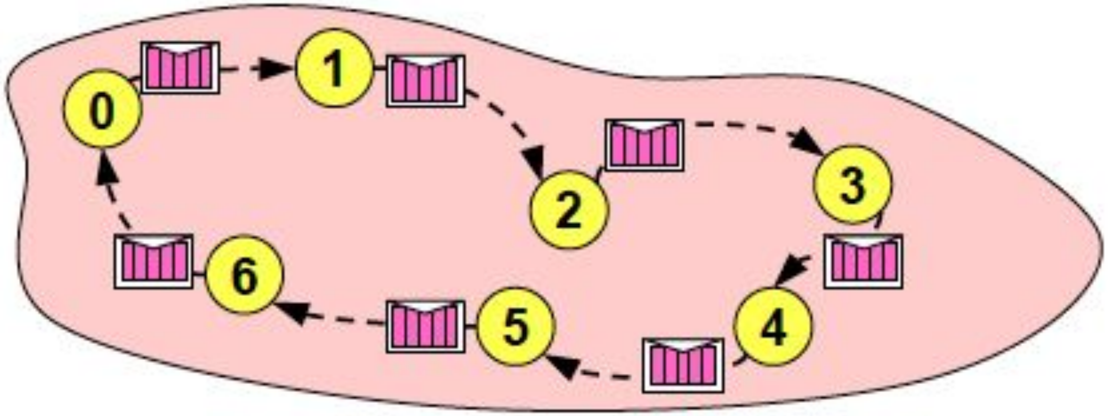
**Non Blocking Communication** permits processes to keep working while checking once in a while if a message has been sent

# Non-blocking Communication

## Deadlock

- Code in each MPI process:

```
MPI_Ssend(…,right_rank,…)
MPI_Recv (…,left_rank,…)
```



will block and never return because MPI_Recv cannot be called in the right-hand MPI process

- Same problem with standard send mode (MPI_Send), if MPI implementation chooses synchronous protocol

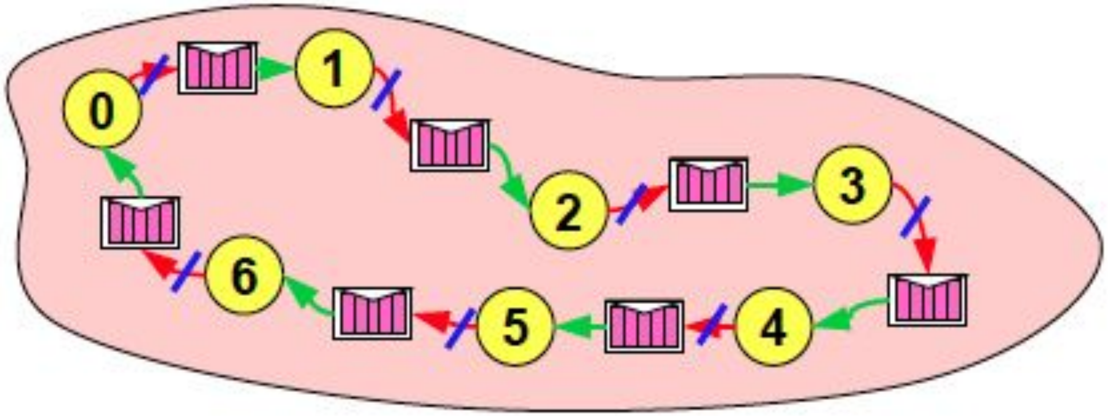
# Non-blocking Communication

Separate communication intro three phases:

- **Initiate nonblocking communication that returns Immediately**
  (routine name starting with **MPI_I**…)
  In the example: Initiate nonblocking send to the right neighbor (□)

- **Do some work**
  In the example: Receiving the message from left neighbor (□)

- **Wait for nonblocking communication to complete** (/)



# Non-blocking Communication

## Nonblocking Send

```
MPI_Isend(void *buf, int count, MPI_Datatype datatype, int
dest, int tag, MPI_Comm comm, MPI_Request *request)

MPI_Wait(MPI_Request *request, MPI_Status *status)
```

- ***buf*** must not be modified between ***Isend*** and ***Wait***
- ***Isend*** **+** ***Wait*** directly after ***Isend*** is equivalent to blocking call (***Ssend***)
- ***status*** is not used.

# Non-blocking Communication

## Nonblocking Receive

```
MPI_Irecv (void *buf, int count, MPI_Datatype datatype, int
source, int tag, MPI_Comm comm, MPI_Request *request)

MPI_Wait(MPI_Request *request, MPI_Status *status)
```

- ***buf*** must not be modified between ***Irecv*** and ***Wait***
- Message ***status*** is returned in ***Wait***
- **Request handle:**
  - Must be stored in local variables: **MPI_Request**
  - Is generated by a nonblocking communication routine
  - Is used (and freed) in the **MPI_WAIT** routine

# Non-blocking Communication

## Blocking and Non-Blocking

- Send and receive can be blocking or nonblocking

- A blocking send can be used with a nonblocking receive, and vice-versa

- Nonblocking sends can use any mode

| | |
|---|---|
| – standard: | MPI_ISEND |
| – synchronous: | MPI_ISSEND |
| – buffered: | MPI_IBSEND |
| – ready: | MPI_IRSEND |

- Synchronous mode affects completion, i.e. MPI_Wait / MPI_Test,

# Non-blocking Communication

## Completion

```
MPI_Wait(MPI_Request *request, MPI_Status *status)

MPI_Test(MPI_Request *request,int *flag, MPI_Status *status)
```

•Completion can be checked by:

- **Wait:** It blocks the task (does not go back of the routine) until the communication has finalized.
- **Test:** It gives back a value TRUE or FALSE depending if the communication has finalized or not.

# Non-blocking Communication

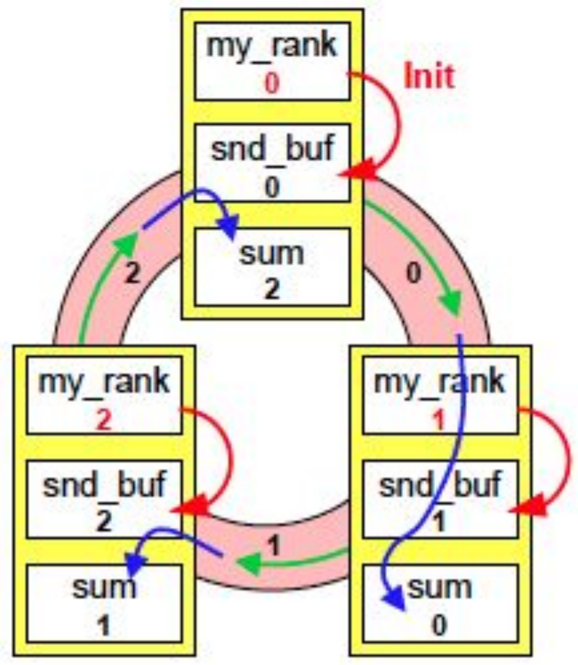
## Example

```
/* Blocking*/
MPI_RECV(x,N,MPI_Datatype,…,&status)

/* Non Blocking with wait */
MPI_IRECV(x,N,MPI_Datatype,…,&request)
… Realise unrelated work with the vector X
MPI_WAIT(&request, &status)
… Realise work related with the vector X

/* Non Blocking with test */
MPI_IRECV(x,N,MPI_Datatype,…,&request)
MPI_TEST(&request, &flag, &status)
while (flag==FALSE) {
     … Realise unrelated work with the vector X
     MPI_TEST(&request, &flag, &status)
}
… Realise work related with the vector X
```

# Non-blocking Communication

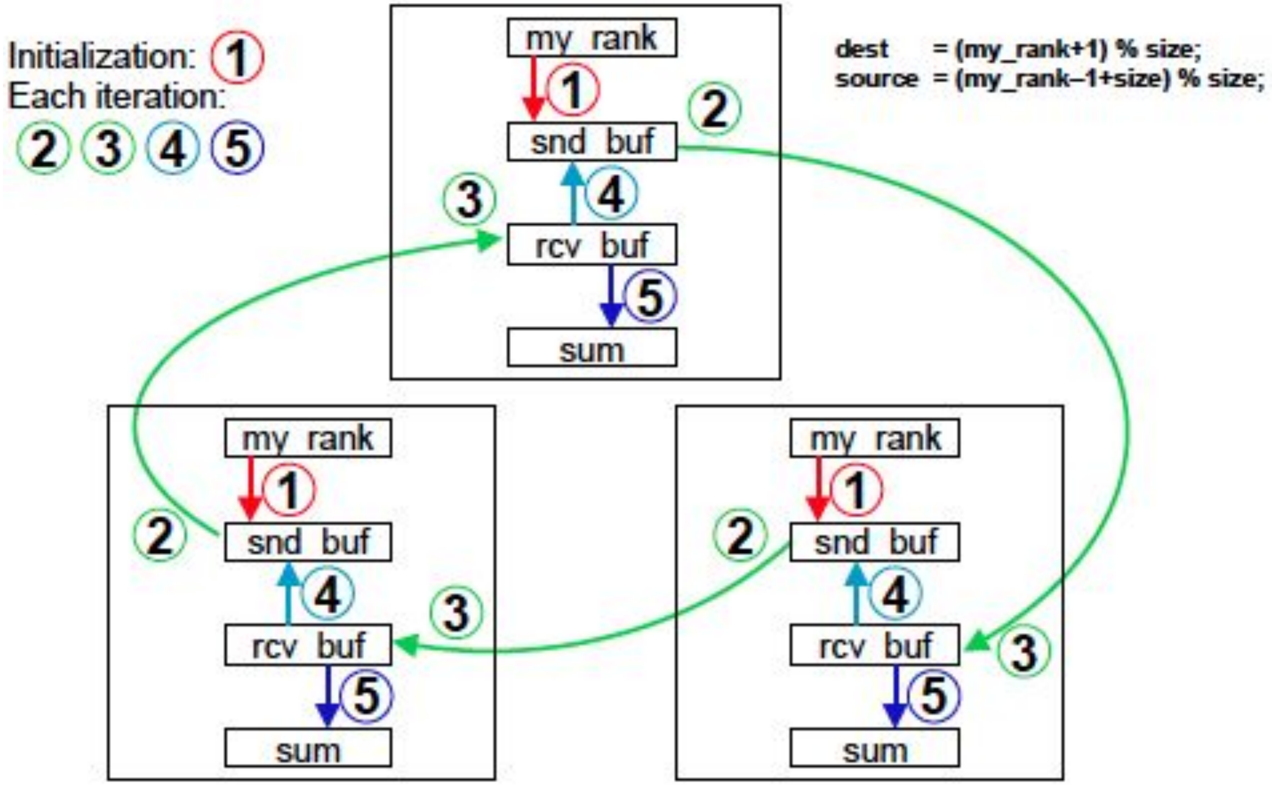
## Activity 4: Rotating information around a ring

1. A set of processes are arranged in a ring.
2. Each process stores its rank into an integer variable *snd_buf*.
3. Each process passes this on to its neighbour on the right.
4. Each processor calculates the sum of the values.
5. Repeat steps 2-5 with "size" iterations (size = number of processes)
6. Implement the program using blocking operations and verify the correctness.
7. Use nonblocking MPI_Issend and verify the correctness.



# Non-blocking Communication

## Activity4: Rotating information around a ring



## Thanks for your atention!

# MPI – Introduction to the Message Passing Interface

CHAPTER 4