# Objective

- MPI point-to-point communication
- Examples
- Hands-on

### MPI\_Barrier

- MPI\_Barrier --> synchronize all processes. Using it frequently in the program increases the computational time
- All collective communication functions have in-built MPI\_Barrier function

### Point-to-point communication

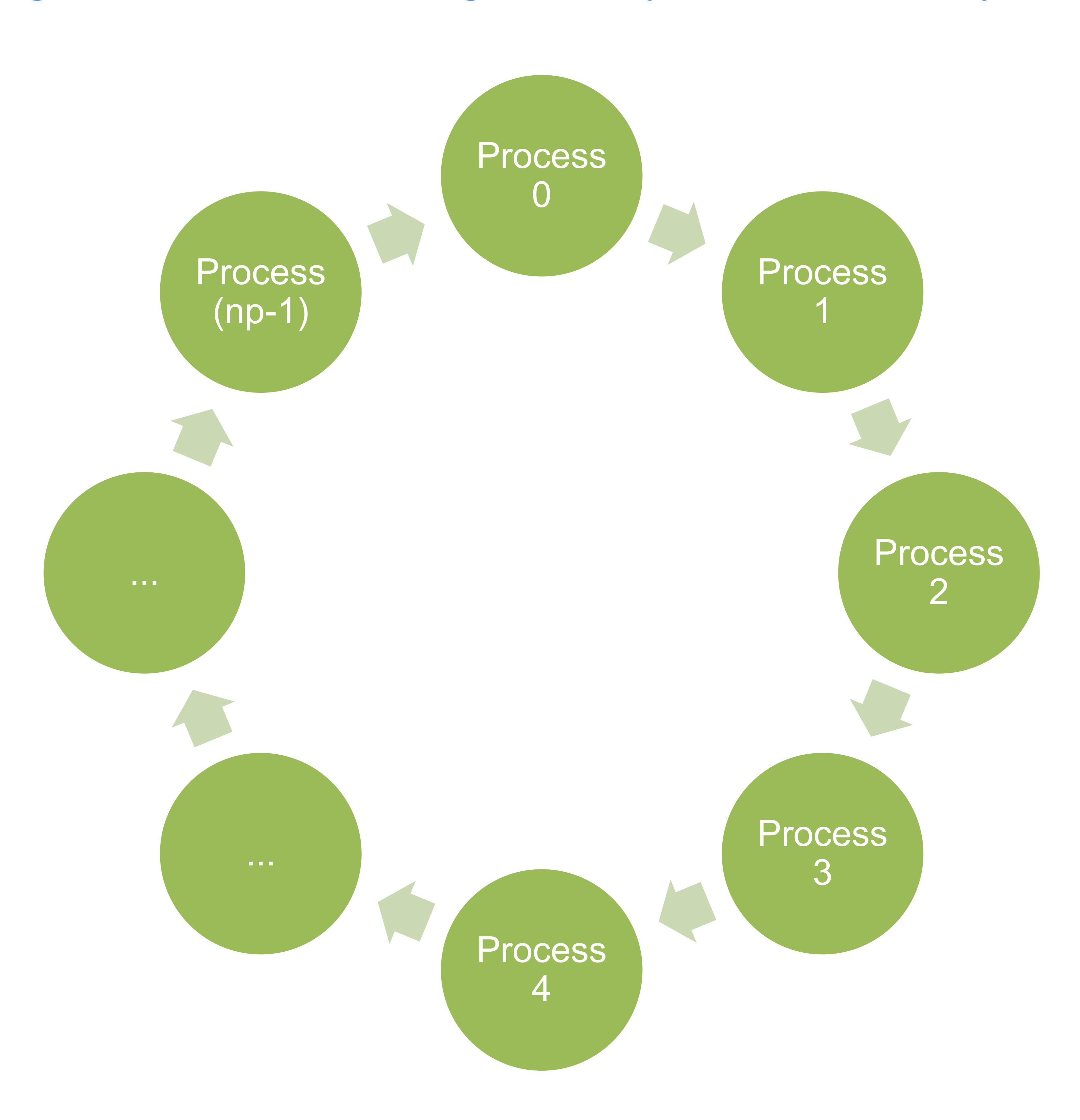
MPI\_Send(data, count, MPI\_datatype, destination\_process\_id, tag,
 MPI Comm, error)

MPI\_Recv(data, count, MPI\_datatype, source\_process\_id, tag,
 MPI Comm, MPI status, error)

Tag (integer number): used to distinguish between different types of messages between two processes (wild card: MPI\_Any\_Tag can be used during reception)

MPI\_status: contains the details of the received message. Usually not required. Use a wild card MPI\_Status\_Ignore during reception

# Sending data in a ring-like pattern/topology



# Ring - MPI\_Send/MPI\_Recv

```
program test
 implicit none
 include 'mpif.h'
integer :: p, id, err, root, msg, tag
 call MPI Init(err)
 call MPI Comm Size(MPI Comm World, p, err)
call MPI Comm Rank(MPI Comm World, id, err)
 root=0; tag=0
 if(id==root) then
  msg=10
   call MPI Send(msg, 1, MPI Int, 1, tag, MPI Comm World, err)
  else
   call MPI Recv(msg, 1, MPI Int, id-1, MPI Any Tag, MPI Comm World, MPI Status Ignore, err)
   write(*,*) id, 'received from process:',id-1
   call MPI Send(msg,1,MPI Int,mod(id+1,p),tag,MPI Comm World,err)
 endif
 if (id==root) then
   call MPI Recv(msg,1,MPI Int,p-1,MPI Any Tag,MPI Comm World,MPI Status Ignore,err)
  write(*,*) id, 'received from process:',p-1
 endif
 call MPI Finalize(err)
end program test
```

### Output

```
mpirun -np 4 ./mpiring.x

1 received from process:
2 received from process:
3 received from process:
0 received from process:
3
```

#### Hands on

- Write the program shown in the previous slides (ring), but without using MPI\_Any\_Tag constant
- Write a program to read 5 numbers from standard input and send them to other processes. Use MPI point-to-point communications.
- Write a program to calculate the sum of first N numbers. Use MPI point-to-point communications