MPI Non-blocking Communication

ICHEC
Irish Centre for High-End Computing

An Roinn Post, Fiontar agus Nuálaíochta
Department of Jobs, Enterprise and Innovation

sfi
science foundation ireland
fondúireacht eolaíochta éireann

AN ROINN
OIDEACHAIS AGUS SCILEANNA
DEPARTMENT OF
EDUCATION AND SKILLS

HEA
Higher Education Authority
An tÚdarás um Ard-Oideachas

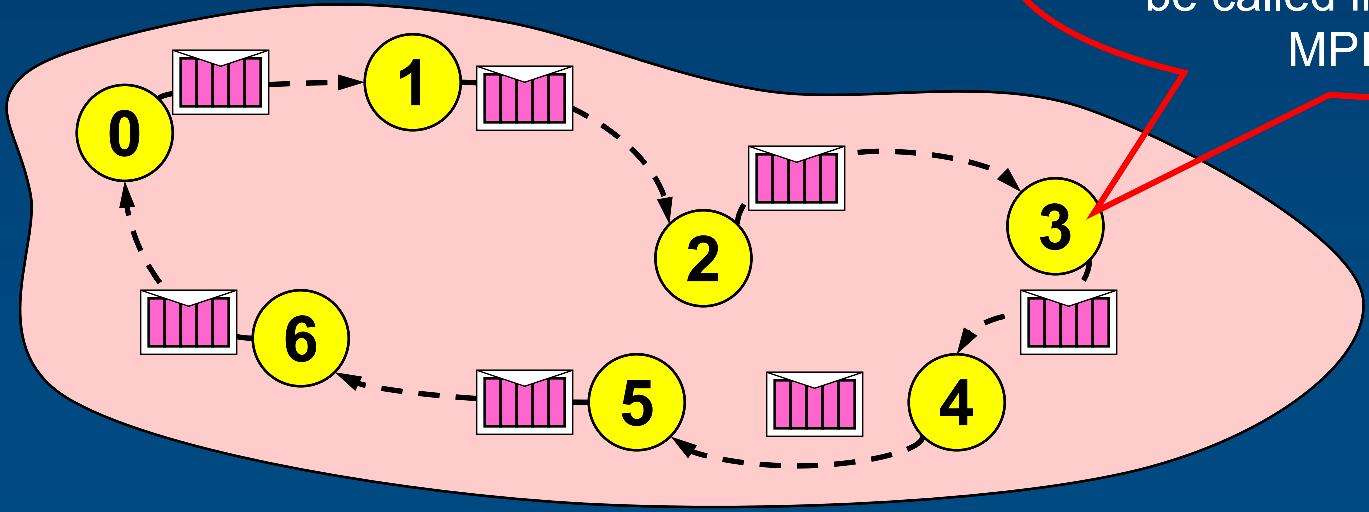
ICHEC
Irish Centre for High-End Computing

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

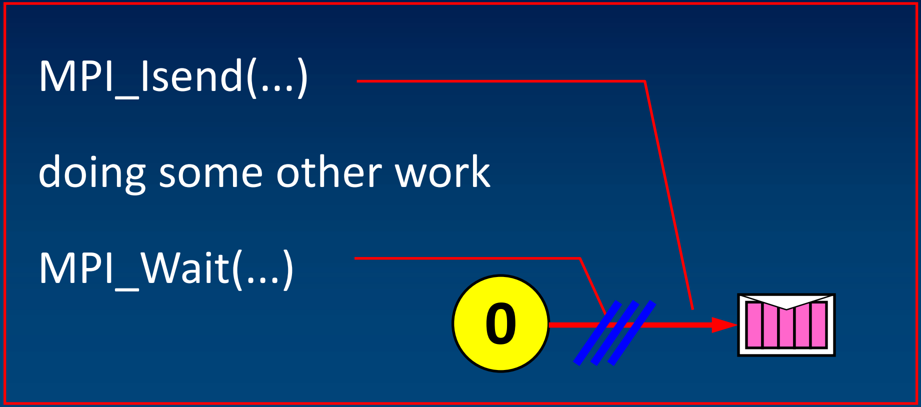
ICHEC
Irish Centre for High-End Computing

Non-Blocking Communications

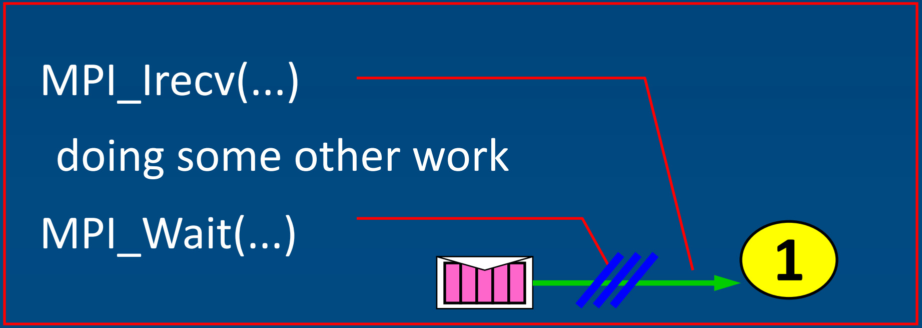
- Separate communication into three phases:
- Initiate non-blocking communication
  - returns **I**mmediately
  - routine name starting with MPI_**I**…
- Do some work
  - "latency hiding"
- Wait for non-blocking communication to complete

Non-Blocking Examples

- Non-blocking **send**



- Non-blocking **receive**

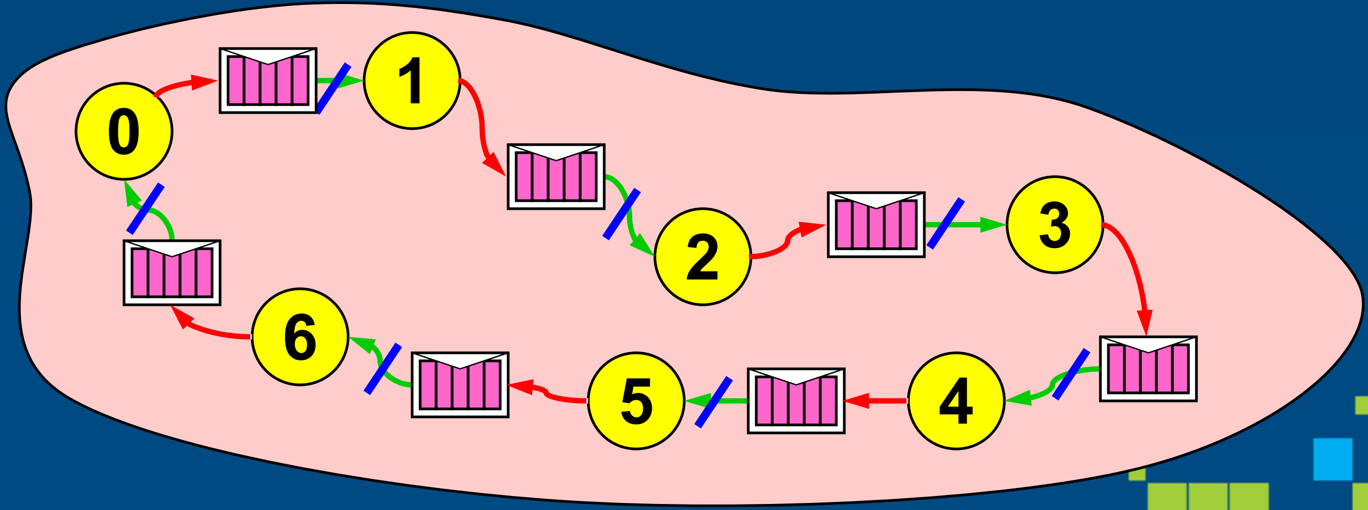


/// = waiting until operation locally completed

ICHEC
Irish Centre for High-End Computing

Non-Blocking Send

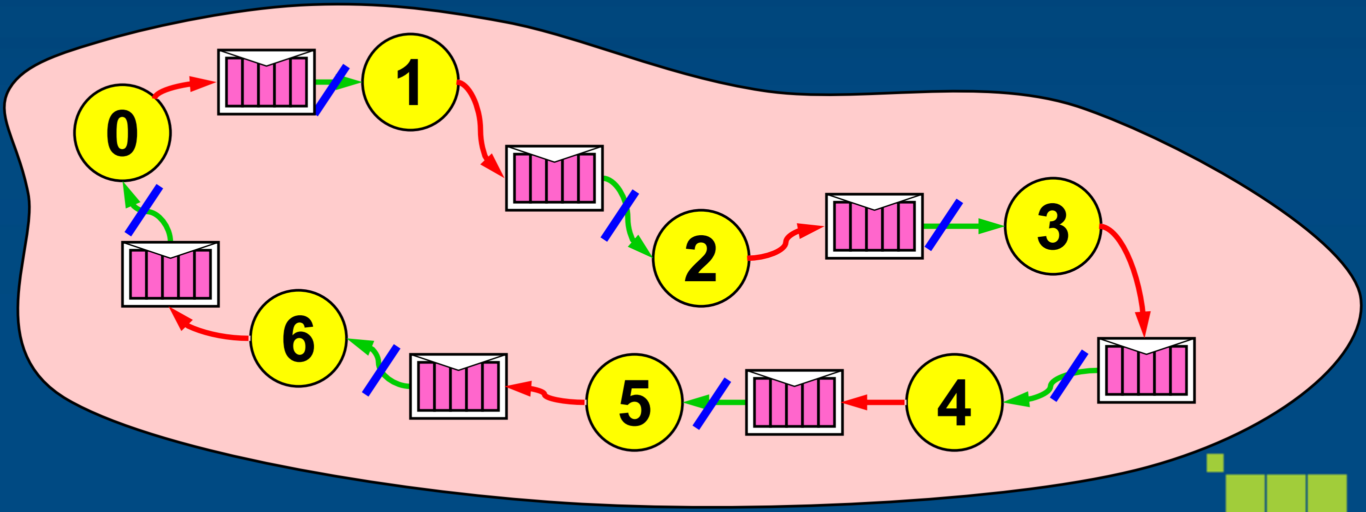
- Initiate non-blocking send
  - ⟶ in the ring example: Initiate non-blocking send to the right neighbor
- Do some work:
  - ⟶ in the ring example: Receiving the message from left neighbor
- Now, the message transfer can be completed
- Wait for non-blocking send to complete /



ICHEC Irish Centre for High-End Computing

Non-Blocking Receive

- Initiate non-blocking receive
  - $\longrightarrow$ in the ring example: Initiate non-blocking receive from left neighbor
- Do some work:
  - $\longrightarrow$ in the ring example: Sending the message to the right neighbor
- Now, the message transfer can be completed
- Wait for non-blocking receive to complete /



Non-blocking Synchronous Send

- C:
  - MPI_Issend(buf, count, datatypeHandle, dest, tag, comm,
    OUT &requestHandle);
  - MPI_Wait(INOUT &requestHandle, &status);
- Fortran:
  - call MPI_Issend(buf, count, datatypeHnadle, dest, tag, comm,
    OUT requestHandle, ierror)
  - call MPI_Wait(INOUT requestHandle, status, ierror)
- Request handle must be stored in local variables
  - C: MPI_Request
  - Fortran: integer
- buf must not be used between Issend and Wait (in all progr. languages)
- "Issend + Wait directly after Issend" is equivalent to blocking call (Ssend)

ICHEC
Irish Centre for High-End Computing

Non-blocking Receive

- C:
  - MPI_Irecv(buf, count, datatypeHandle, source, tag, comm, OUT &requestHandle);
  - MPI_Wait(INOUT &requestHandle, &status);
- Fortran:
  - call MPI_Irecv (buf, count, datatype, source, tag, comm, OUT requestHandle, ierror)
  - call MPI_Wait(INOUT requestHandle, status, ierror)
- buf must not be used between Irecv and Wait (in all progr. languages)

Blocking and Non-Blocking

- Send and receive can be blocking or non-blocking.

Send Mode	Blocking Function	Nonblocking Function
Standard	MPI_Send	MPI_Isend
Synchronous	MPI_Ssend	MPI_Issend
Ready	MPI_Rsend	MPI_Irsend
Buffered	MPI_Bsend	MPI_Ibsend

ICHEC
Irish Centre for High-End Computing

Completion

- C:
  - MPI_Wait(&requestHandle, &status);
  - MPI_Test(&requestHandle, &flag, &status);
- Fortran:
  - call MPI_Wait(requestHandle, status, ierror)
  - call MPI_Test(requestHandle, flag, status, ierror)
- one must
  - wait or
  - loop with TEST until request is completed, i.e., flag == 1 or .true.

ICHEC Irish Centre for High-End Computing

Multiple Non-Blocking Communications

- You have several request handles:
- Wait or test for completion of one message
  - MPI_Waitany / MPI_Testany
- Wait or test for completion of all messages
  - MPI_Waitall / MPI_Testall
- Wait or test for completion of as many messages as possible
  - MPI_Waitsome / MPI_Testsome