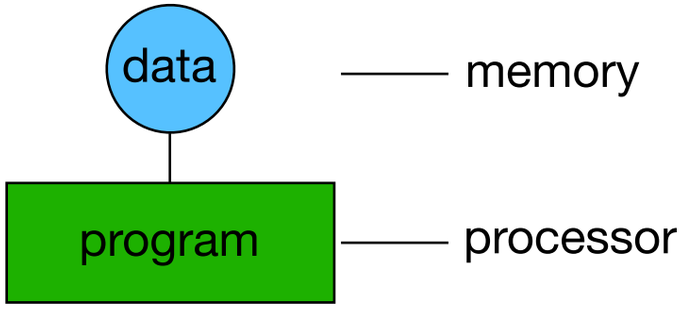
Introduction to Parallel Programming 2

Teeraparb Chantavat

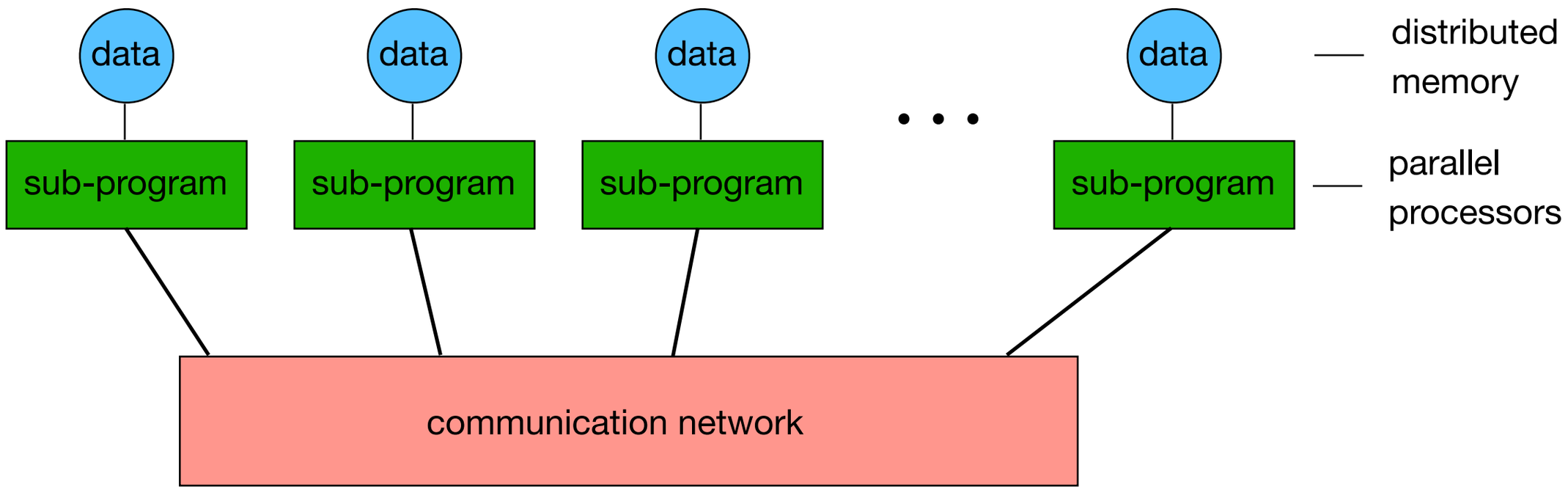
Institute for Fundamental Study
Naresuan University

The Message-Passing Programming Paradigm

- **Sequential Programming Paradigm**

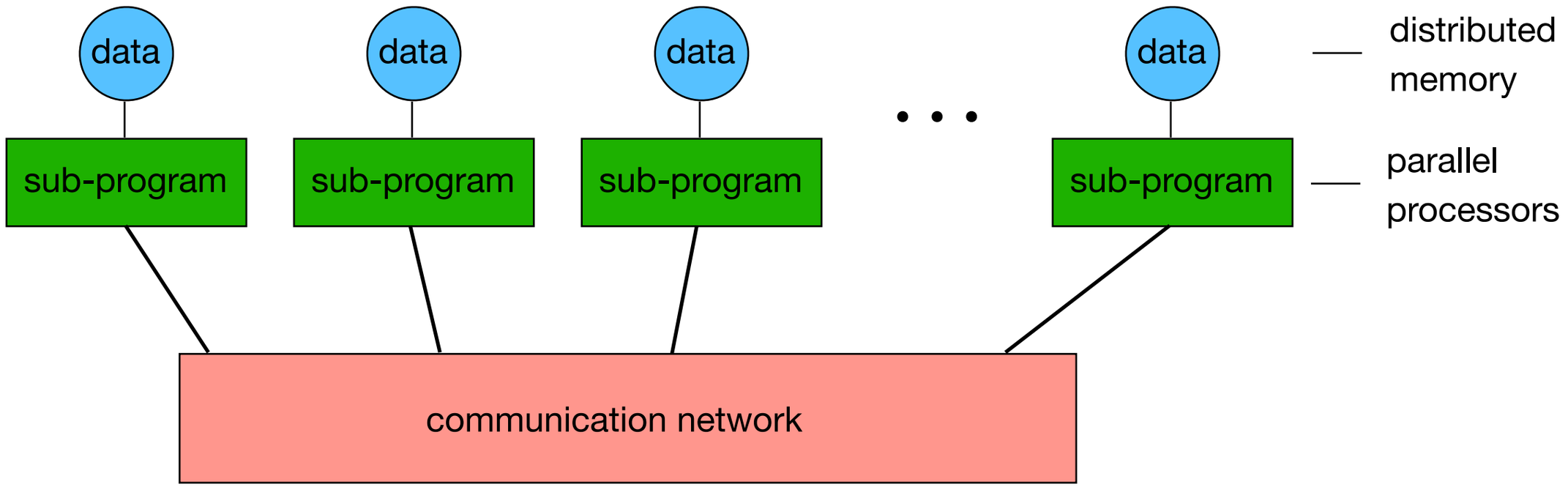


- **Message-Passing Programming Paradigm**



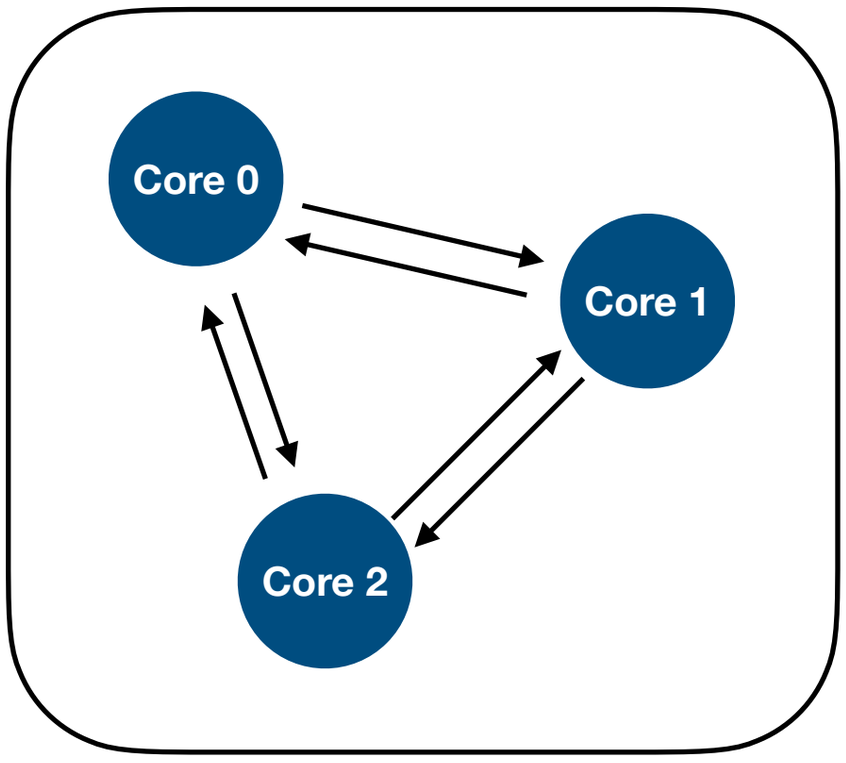
The Message-Passing Programming Paradigm

- Each processor in a message passing program
  - written in a conventional sequential language, e.g., C/C++, Fortran, or Python
  - typically the same on each processor (SPMD)
  - the variables of each sub-program have
  - communicate via
    - **the same name**
    - **but different locations (distributed memory) and different data!**
      - **i.e. all variables are private**
  - communicate via special send & receive routines (***message passing***)

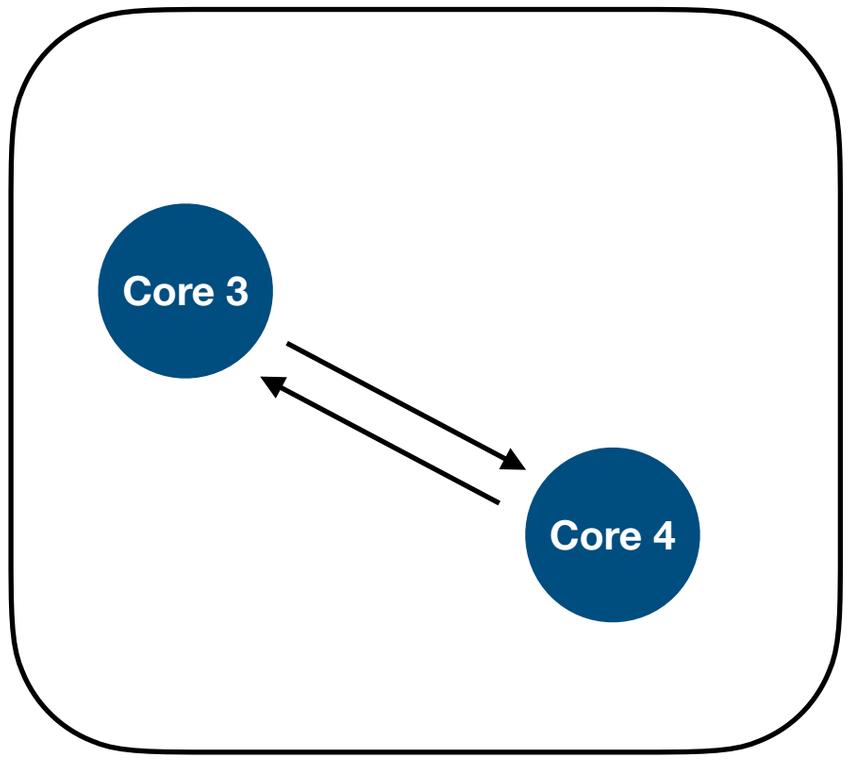


Communicators

A **communicator** is a group of cores that can communicate to one another. Each cores is independent and can run its own process.



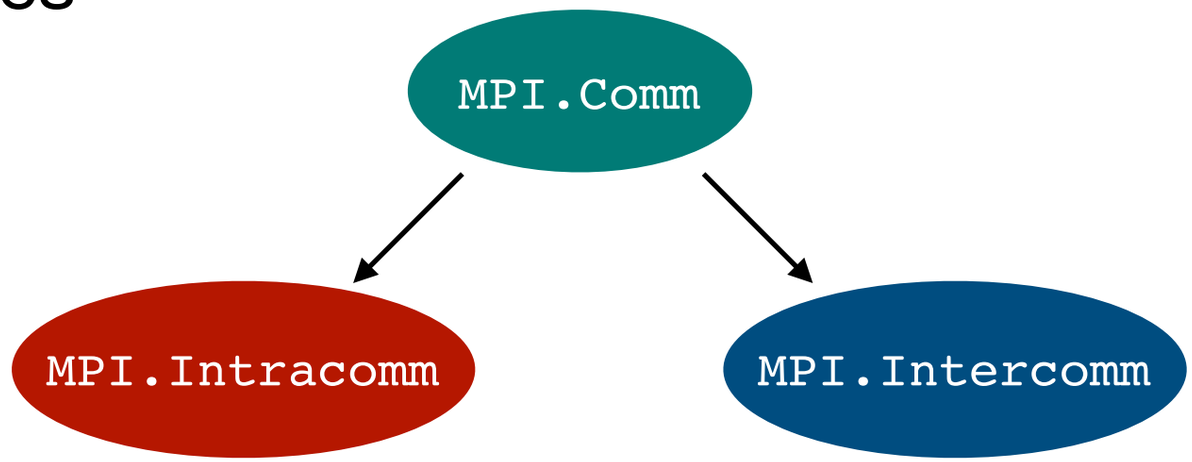
Communicator A



Communicator B

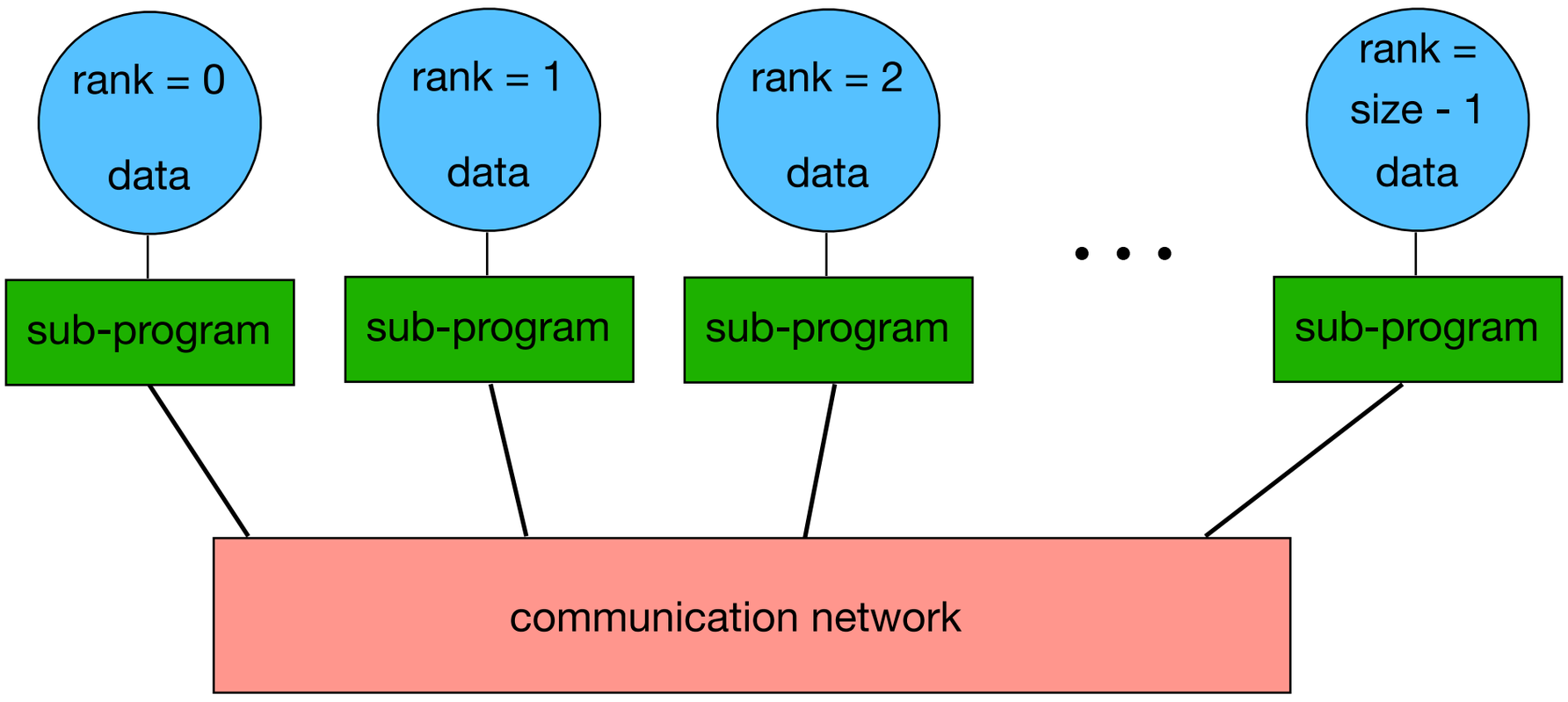
Communicators

- In *MPI for Python*, `MPI.Comm` is the base class of communicators.
- `MPI.Intracomm` is a sub-class of `MPI.Comm` mainly used for local nodes. **`MPI.COMM_WORLD`** and **`MPI.COMM_SELF`** are pre-defined instances of `MPI.Intracomm`.
- `MPI.Intercomm` is also a sub-class of `MPI.Comm` mainly used for remote nodes



Data and Work Distribution

- The value of **rank** is returned by special library routine.
- The system of **size** processes is started by special MPI initialization.
- All distribution decision are based on rank
- i.e., which process work on which data



See MPI/ex00_mpi_greeting.py

Point-to-point Communications

- Simplest form of message passing.
- One process (core) sends a message to another. Another process (core) will receive the message

`MPI.Comm.Send` `MPI.Comm.Recv`

```
MPI.Comm.Send(buf, dest, tag=0)
```

```
MPI.Comm.Recv(buf, source=ANY_SOURCE, tag=ANY_TAG,
              status=None)
```

See MPI/ex01_mpi_send_recv0.py

MPI DataType

- When sending an array of data, the type of MPI needs to know the type of data.
- Sometime the data type has to be specified in the data buffer.

`MPI.CHAR`	`MPI.LONG`
`MPI.BYTE`	`MPI.FLOAT`
`MPI.SHORT`	`MPI.DOUBLE`
`MPI.INT`	`MPI.COMPLEX`

See MPI/ex02_mpi_send_recv1.py

MPI for Python object

- On top of Standard MPI commands, `mpi4py` has commands for generic Python objects using all lowercases.

- Python objects are serialized / deserialized using `pickle` module

```
MPI.Comm.send(object, dest, tag=0)
```

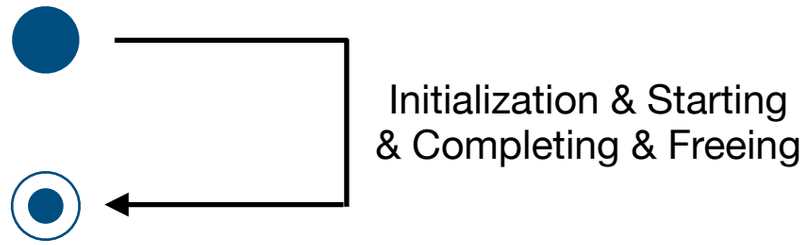
```
MPI.Comm.recv(object, source=ANY_SOURCE, tag=ANY_TAG,
              status=None)
```

See MPI/ex03_mpi_send_recv2.py

MPI Operations

- **Blocking**

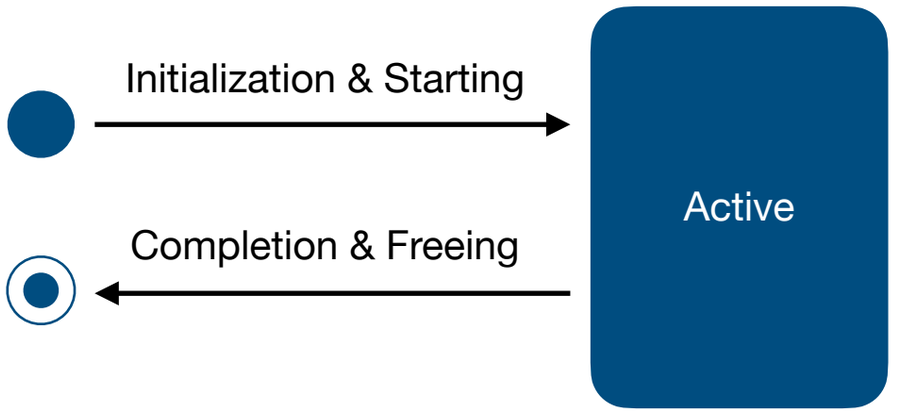
  Initialization & Starting & Completion & Freeing



- **Non-blocking**

  Initialization & Starting

  Completion & Freeing



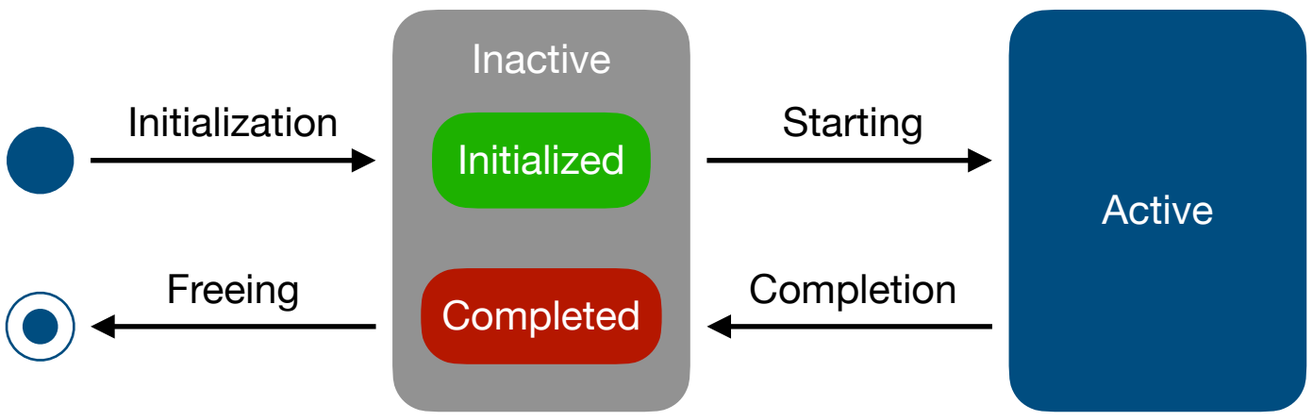
- **Persistent**

  Initialization

  Starting

  Completion

  Freeing



See MPI/ex04_send_recv3.py

MPI Send Communication Modes

`MPI_Send` has 4 modes of communication while `MPI_Recv` has only 1 mode.
Both can be blocking or non-blocking.

- **Standard**
  The MPI library decide whether or not the non-local buffered the outgoing data.
- **Buffered (Asynchronous)**
  The MPI library decide to use buffer for outgoing data if no matching receiver has been posted.
- **Synchronous**
  The outgoing data buffer can be reused once the receiving process starting receiving the data.
- **Ready**
  The outgoing data buffer can be reused once the receiving process has been posted.

MPI Send Variants

Communication Mode	Blocking	Non-blocking	Persistent
Standard	`MPI.Comm.Send` `MPI.Comm.send`	`MPI.Comm.Isend` `MPI.Comm.isend`	`MPI.Comm.Send_Init`
Buffered (Asynchronous)	`MPI.Comm.Bsend` `MPI.Comm.bsend`	`MPI.Comm.Ibsend` `MPI.Comm.ibsend`	`MPI.Comm.BSend_Init`
Synchronous	`MPI.Comm.Ssend` `MPI.Comm.ssend`	`MPI.Comm.ISend` `MPI.Comm.isend`	
Ready	`MPI.Comm.Rsend`	`MPI.Comm.Irsend`	

for generic Python objects

MPI Recv Variants

Communication Mode	Blocking	Non-blocking	Persistent
Standard	`MPI.Comm.Recv` `MPI.Comm.recv`	`MPI.Comm.Irecv` `MPI.Comm.irecv`	`MPI.Comm.Recv_Init`

for generic Python objects

MPI Sendrecv Variants

- The **send-receive** operation combine in one operation the sending of a message to one destination and the receiving of another message, from another process

Communication Mode	Blocking

Standard	`MPI.Comm.Sendrecv`
	`MPI.Comm.Sendrecv_replace`
	`MPI.Comm.sendrecv`

See MPI/ex05_mpi_send_recv4.py