

# HPC.NRW

# MPI in Small Bites

## PPCES 2025

HPC.NRW Competence Network



**EDIH**  
Rheinland



THE COMPETENCE NETWORK FOR HIGH PERFORMANCE COMPUTING IN NRW.

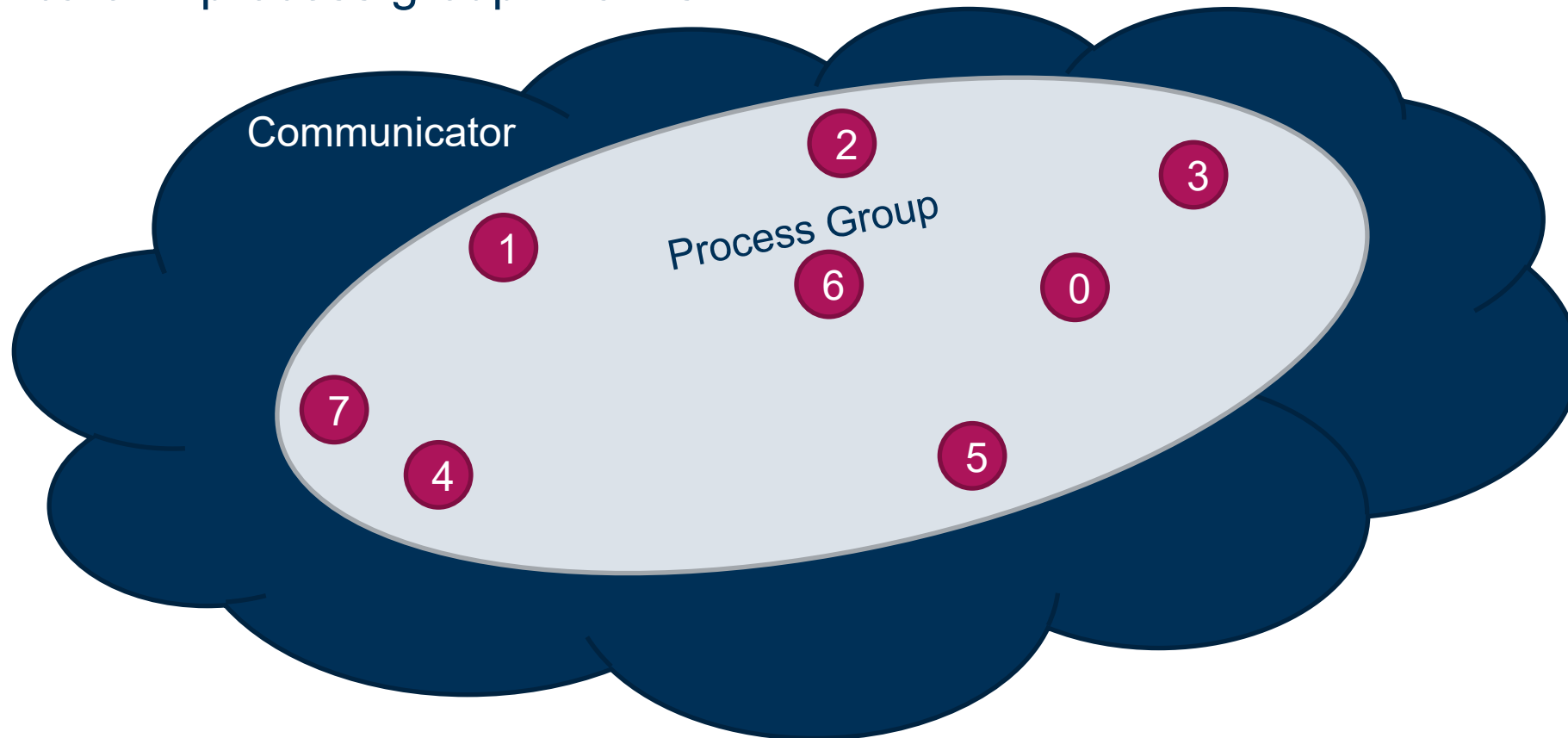
# Communicator and Group Handling

HPC.NRW Competence Network

## MPI in Small Bites

- Defines context for each communication operation in MPI
  - Group of participating peers (process group)
  - Error handlers for communication and I/O operations
  - Local key/value cache
  - Virtual topology information (optional)
- Two types: intra-communicators (single world) and inter-communicators (across worlds)
  - Inter-communicators not covered here (→ Dynamic Process Management)
- Two predefined intra-communicators (pre MPI 4.0 and MPI 4.0 World Model):
  - **MPI\_COMM\_WORLD**  
contains all processes launched **initially** as part of the MPI program
  - **MPI\_COMM\_SELF**  
contains only the current process

- Communicator – process group – ranks



- Obtain the size of the process group of a given communicator:

```
MPI_Comm_size (MPI_Comm comm, int *size)
```

- Ranks in the group are numbered from 0 to size-1
- Obtain the rank of the calling process in the given communicator:

```
MPI_Comm_rank (MPI_Comm comm, int *rank)
```

- Special “null” rank – MPI\_PROC\_NULL
  - Can be source or destination of point-to-point communications
    - Corresponding communication call transforms into a no-op and returns immediately
  - Used to write symmetric code and handle process boundaries

- Comparing handles directly has limited value
  - No information about the opaque objects behind the handles

```
MPI_Comm_compare (MPI_Comm comm1, MPI_Comm comm2, int *result)
```

- Result can be:
  - **MPI\_IDENT**
    - The communicators are identical (i.e., `comm1 == comm2`)
  - **MPI\_CONGRUENT**
    - The underlying groups are identical in constituents and rank order, but the context is different (e.g., after duplication)
  - **MPI\_SIMILAR**
    - The group members are the same, but in different order
  - **MPI\_UNEQUAL**
    - Otherwise

- Duplicate an existing communicator
  - `MPI_Comm_dup`, `MPI_Comm_dup_with_info`, `MPI_Comm_idup`
- Create new communicator for a subgroup of a communicator
  - `MPI_Comm_create`, `MPI_Comm_create_group`
- Split an existing communicator
  - `MPI_Comm_split`, `MPI_Comm_split_type`

- Duplicate a given communicator:

```
MPI_Comm_dup (MPI_Comm comm, MPI_Comm *newcomm)
```

- New communication context with same ranks and ordering
- Easy isolation of encapsulated communication
  - Libraries should never communicate on MPI\_COMM\_WORLD directly
- Potentially modified info settings are not duplicated
  - MPI\_Comm\_dup\_with\_info
- Communicator creation can be costly
  - Nonblocking versions available



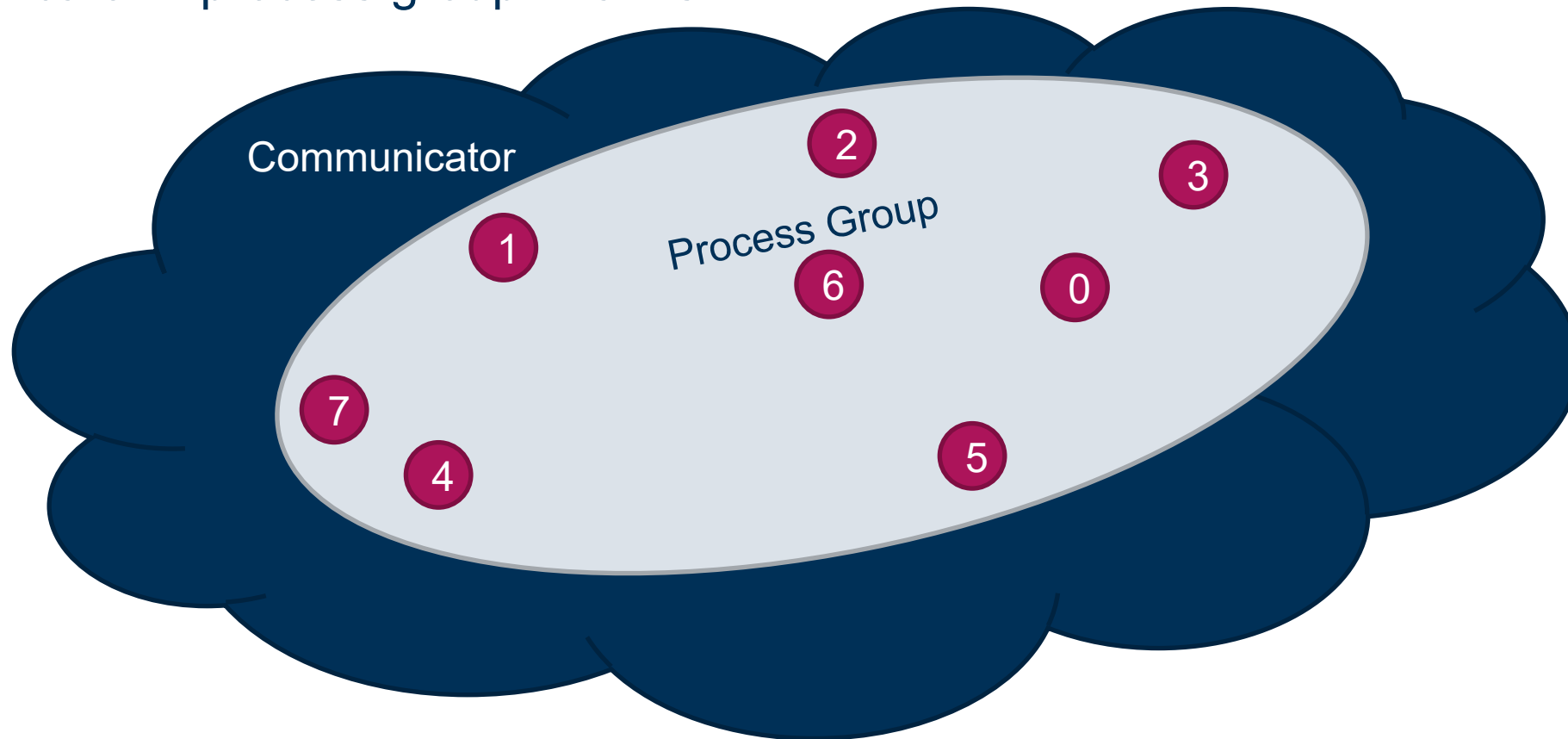
- Communicators take up memory and other precious resources
- Should be freed once no longer needed

```
MPI_Comm_free (MPI_Comm *comm)
```

- Marks **comm** for deletion
  - **comm** is set to **MPI\_COMM\_NULL** on return
  - The actual communicator object is only deleted once all pending operations are completed
- 
- It is erroneous to free predefined communicators **MPI\_COMM\_WORLD**, **MPI\_COMM\_SELF** or **MPI\_COMM\_NULL**

- Duplicate an existing communicator
  - `MPI_Comm_dup`, `MPI_Comm_dup_with_info`
  - `MPI_Comm_idup`, `MPI_Comm_idup_with_info` (since MPI 4.0)
- Create new communicator for a subgroup of a communicator
  - `MPI_Comm_create`, `MPI_Comm_create_group`
- Split an existing communicator
  - `MPI_Comm_split`, `MPI_Comm_split_type`

- Communicator – process group – ranks



- Ordered set of processes
  - Rank is actually a characteristic of the communicator's underlying group
- MPI processes can be part of different groups
- Multiple communicators can be based on the same group

- Obtain the size of a process group:

```
MPI_Group_size (MPI_Group group, int *size)
```

- ranks in the group are numbered from 0 to size-1

- Obtain the rank of the calling process in the given process group:

```
MPI_Group_rank (MPI_Group group, int *rank)
```

- Comparing handles directly has limited value
  - No information about the opaque objects behind the handles

```
MPI_Group_compare (MPI_Group group1, MPI_Group group2, int *result)
```

- Result can be:
  - **MPI\_IDENT**
    - The groups are identical (i.e., `comm1 == comm2`)
    - The underlying groups are identical in constituents and rank order
  - **MPI\_SIMILAR**
    - The group members are the same, but in different order
  - **MPI\_UNEQUAL**
    - Otherwise

```
MPI_Group_translate_ranks (MPI_Group group1, int n, const int ranks1[],  
                           MPI_Group group2, int rank2[])
```

- **n** indicates the length of the two arrays **ranks1** and **rank2**
- **ranks1** holds a list of valid ranks in **group1**
- **ranks2** returns the corresponding rank in **group2** at the same index
  - **MPI\_UNDEFINED** if no correspondence exists

- No mechanism to build a group from scratch
  - Groups need to be derived from a base group
- Obtain the group of a given communicator

```
MPI_Comm_group (MPI_Comm comm, MPI_Group *group)
```

- Obtain the group of predefined communicator `MPI_COMM_WORLD` and derive from it



- Build unions and or intersections of the process groups

```
MPI_Group_union (MPI_Group group1, MPI_Group group2, MPI_Group *newgroup)  
MPI_Group_intersection (MPI_Group group1, MPI_Group group2, MPI_Group *newgroup)
```

- Remove ranks of a second group from those present in a first group

```
MPI_Group_difference (MPI_Group group1, MPI_Group group2, MPI_Group *newgroup)
```

- Explicitly list ranks to retain in (or remove from) a given group

```
MPI_Group_incl (MPI_Group group, int n, const int ranks[], MPI_Group *newgroup)  
MPI_Group_excl (MPI_Group group, int n, const int ranks[], MPI_Group *newgroup)  
MPI_Group_range_incl (MPI_Group group, int n, const int ranks[][3], MPI_Group  
*newgroup)  
MPI_Group_range_excl (MPI_Group group, int n, const int ranks[][3], MPI_Group  
*newgroup)
```

- Groups take up memory and other precious resources
- Should be freed once no longer needed

```
MPI_Group_free (MPI_Group *group)
```

- Marks **group** for deletion
- **group** is set to **MPI\_GROUP\_NULL** on return
- The actual group object is only deleted once all internal references are released

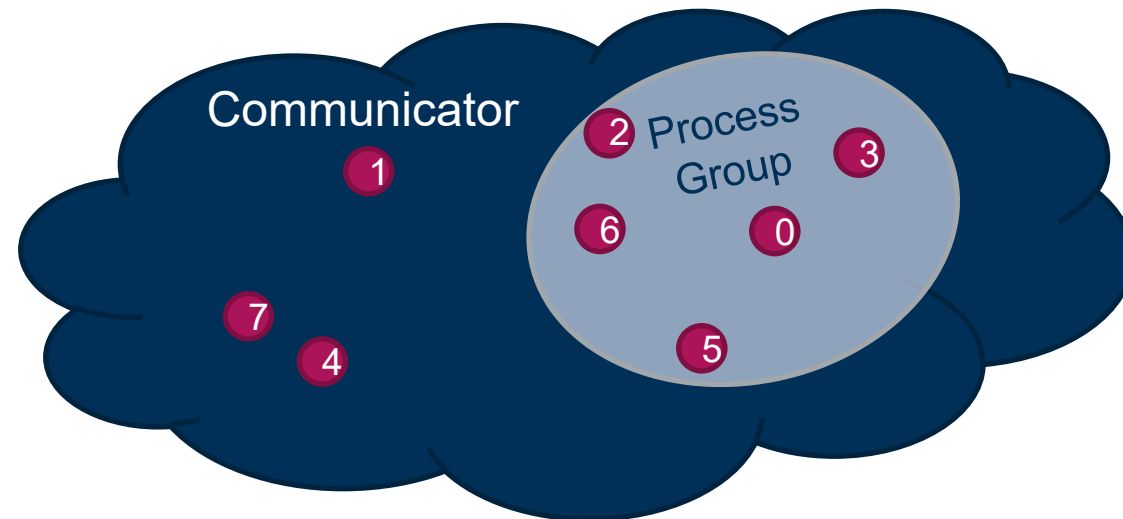
- Create new communicator for a subgroup of a communicator

```
MPI_Comm_create (MPI_Comm comm, MPI_Group group, MPI_Comm *newcomm)
```

- Collective in comm (for ranks group: newcomm=MPI\_COMM\_NULL)

```
MPI_Comm_create_group (MPI_Comm comm, MPI_Group group, int tag,  
MPI_Comm *newcomm)
```

- Collective in group



- Duplicate an existing communicator
  - `MPI_Comm_dup`, `MPI_Comm_dup_with_info`
  - `MPI_Comm_idup`, `MPI_Comm_idup_with_info` (since MPI 4.0)
- Create new communicator for a subgroup of a communicator
  - `MPI_Comm_create`, `MPI_Comm_create_group`
- Split an existing communicator
  - `MPI_Comm_split`, `MPI_Comm_split_type`

- Split existing communicators into parts

```
MPI_Comm_split (MPI_Comm comm, int color, int key, MPI_Info info, MPI_Comm *newcomm)
```

- Split by some characteristics (e.g., rank % n, rank < n, rank / n)

```
MPI_Comm_split_type (MPI_Comm comm, int split_type, int key, MPI_Info info,  
MPI_Comm *newcomm)
```

- Split into shared memory groups
- key controls the rank order within newcomm
- Useful for shared memory windows
- One-sided communication

