Defining Communicators

ICHEC
Irish Centre for High-End Computing

ICHEC
Irish Centre for High-End Computing

Communicators

- A communicator determines the "communication universe"
  - Source and destination of a message is identified by the process rank within the communicator
  - MPI_COMM_WORLD used so far
- Processes can be divided into subcommunicators also known as groups
  - Task level parallelism with process 'groups' performing separate duties together
  - Parallel I/O
  - Scalability → avoids unnecessary synchronization

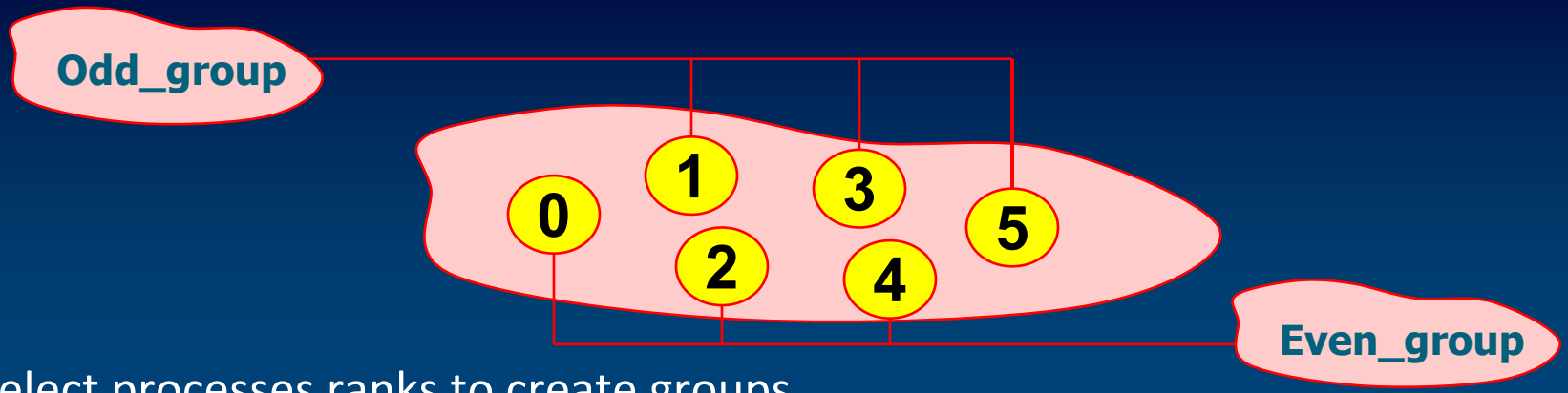
ICHEC
Irish Centre for High-End Computing

Motivations

- Need to create sets of processes
  - For programming convenience
  - Make use of collectives routines
- Need to map the abstract topology onto the natural topology of the problem domain
  - For programming convenience
  - For performance

ICHEC
Irish Centre for High-End Computing

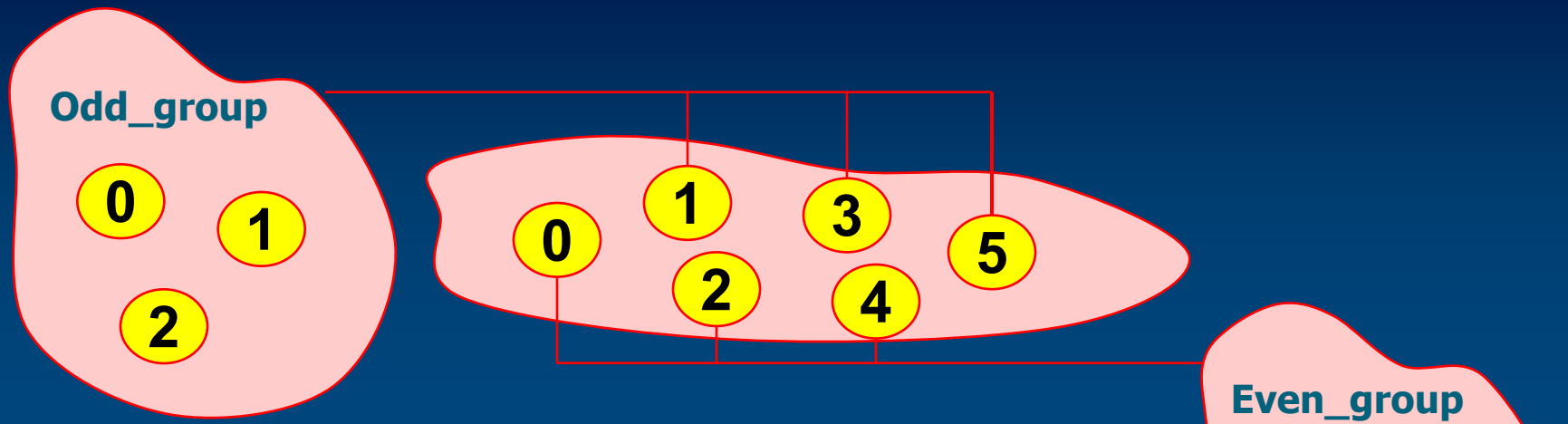
Working with groups



- Select processes ranks to create groups
- Associate to these groups *new* communicators
- Use these new communicators as usual
- MPI_Comm_group(comm, group) returns in *group* the group associated to the communicator *comm*
- Communicators are dynamic and can be created and destroyed during runtime
- A task can belong simultaneously to several communicators, and in each, has a unique ID

ICHEC
Irish Centre for High-End Computing

Working with groups



- NB: Rank numbers restart from 0 in the new communicators
  - Each group is an ordered set of process identifiers
  - Each process in a group is associated with a rank

- WORLD {1, 3, 5} → Odd_group {0, 1, 2}
- WORLD {0, 2, 4} → Even_group {0, 1, 2}

ICHEC
Irish Centre for High-End Computing

Method 1: MPI_comm_group

```
Odd_ranks={1, 3, 5}, Even_ranks={0, 2, 4};
MPI_comm_group(MPI_COMM_WORLD,Group_all);
MPI_Group_incl(Group_all, 3, Odd_ranks, &Odd_group);
MPI_Group_incl(Group_all, 3, Even_ranks, &Even_group);
int MPI_Comm_create(MPI_COMM_WORLD, Odd_group, Odd_Comm);
int MPI_Comm_create(MPI_COMM_WORLD, Even_group, Even_Comm);
```

ICHEC
Irish Centre for High-End Computing

Method 2: MPI_Comm_split

- Create new communicators based on 'colors' and 'keys'

MPI_Comm_split(comm, color, key, &newcomm)

comm	Communicator handle
color	Control of subset assignment, processes with the same color belong to the same new communicator
key	Control of rank assignment
newcomm	New communicator handle

- If color = MPI_UNDEFINED, a process does not belong to any of the new communicators

ICHEC
Irish Centre for High-End Computing

Creating a communicator

```
if (myid%2 == 0) {
    color = 1;
}   else {
    color = 2;
}
MPI_Comm_split(MPI_COMM_WORLD, color, myid, &subcomm);
MPI_Comm_rank(subcomm, %mysubid);
printf("I am rank %d in MPI_COMM_WORLD, but %d in Comm %d.\n", myid,
mysubid, color);
```

```
I am rank 2 in MPI_COMM_WORLD, but 1 in Comm 1.
I am rank 1 in MPI_COMM_WORLD, but 1 in Comm 2.
I am rank 4 in MPI_COMM_WORLD, but 1 in Comm 1.
I am rank 5 in MPI_COMM_WORLD, but 1 in Comm 2.
I am rank 3 in MPI_COMM_WORLD, but 1 in Comm 2.
I am rank 0 in MPI_COMM_WORLD, but 1 in Comm 1.
```

ICHEC
Irish Centre for High-End Computing

Group & Communicator Management

Group	Communicator
Accessors	
`MPI_Group_size(…)`	`MPI_Comm_size(…)`
`MPI_Group_rank(…)`	`MPI_Comm_rank(…)`
…	…
Constructors	
`MPI_Comm_group(…)`	`MPI_Comm_create(…)`
`MPI_Group_incl(…)`	`MPI_Comm_split(…)`
`MPI_GROUP_EXCL(…)`	
…	
Destructors	
`MPI_Group_free(group)`	`MPI_Comm_free(comm)`