Communicators

Feb 5, 2021

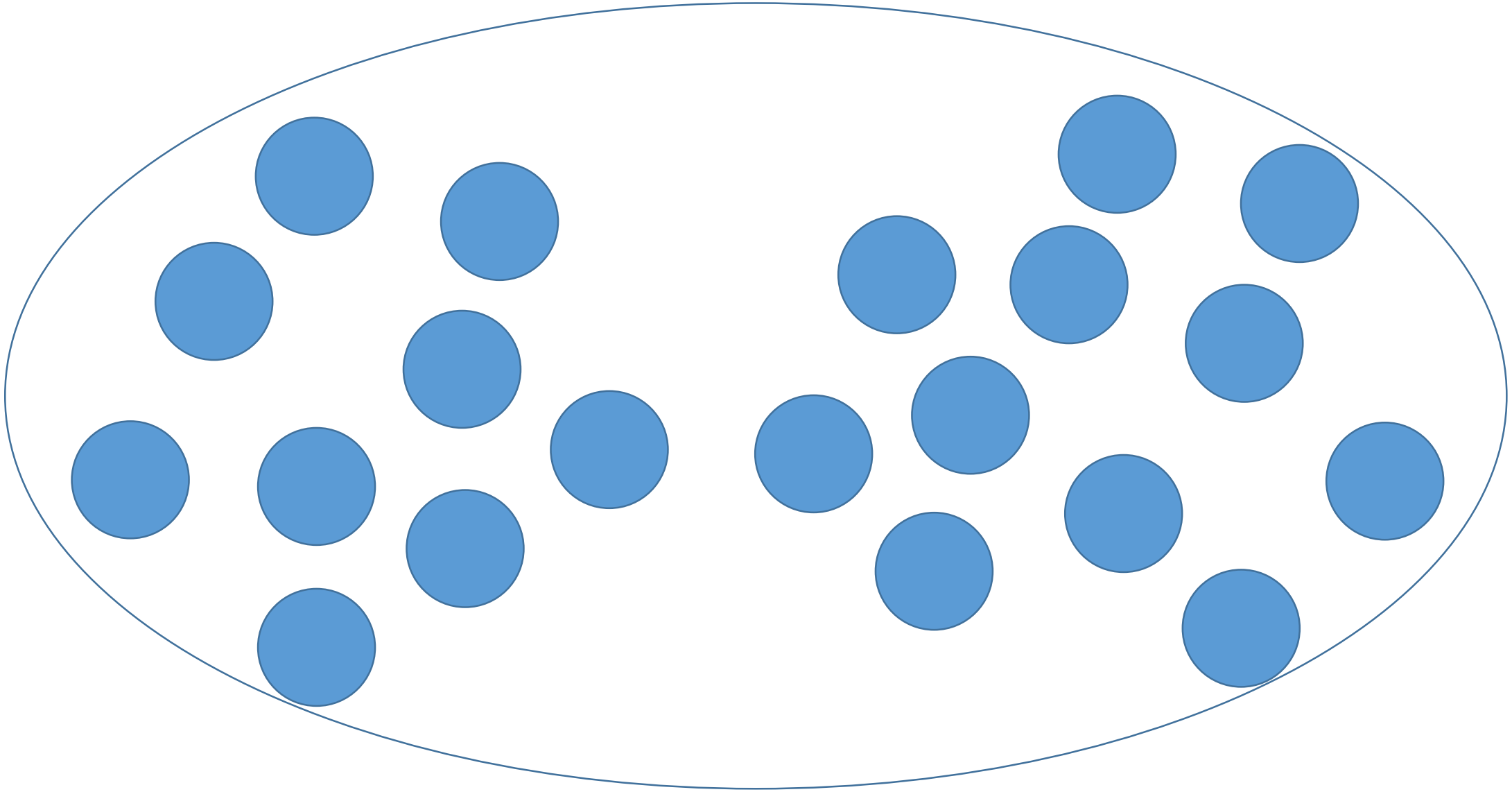
Communicator

- Object containing a group of processes
- Representative of communication domain
- Predefined:
  - MPI_COMM_WORLD
  - MPI_COMM_SELF
- Contains a mapping from MPI process ranks to processor ids
- Memory proportional to #processes in the group
- Several communication contexts may co-exist within a single communicator

Process Group

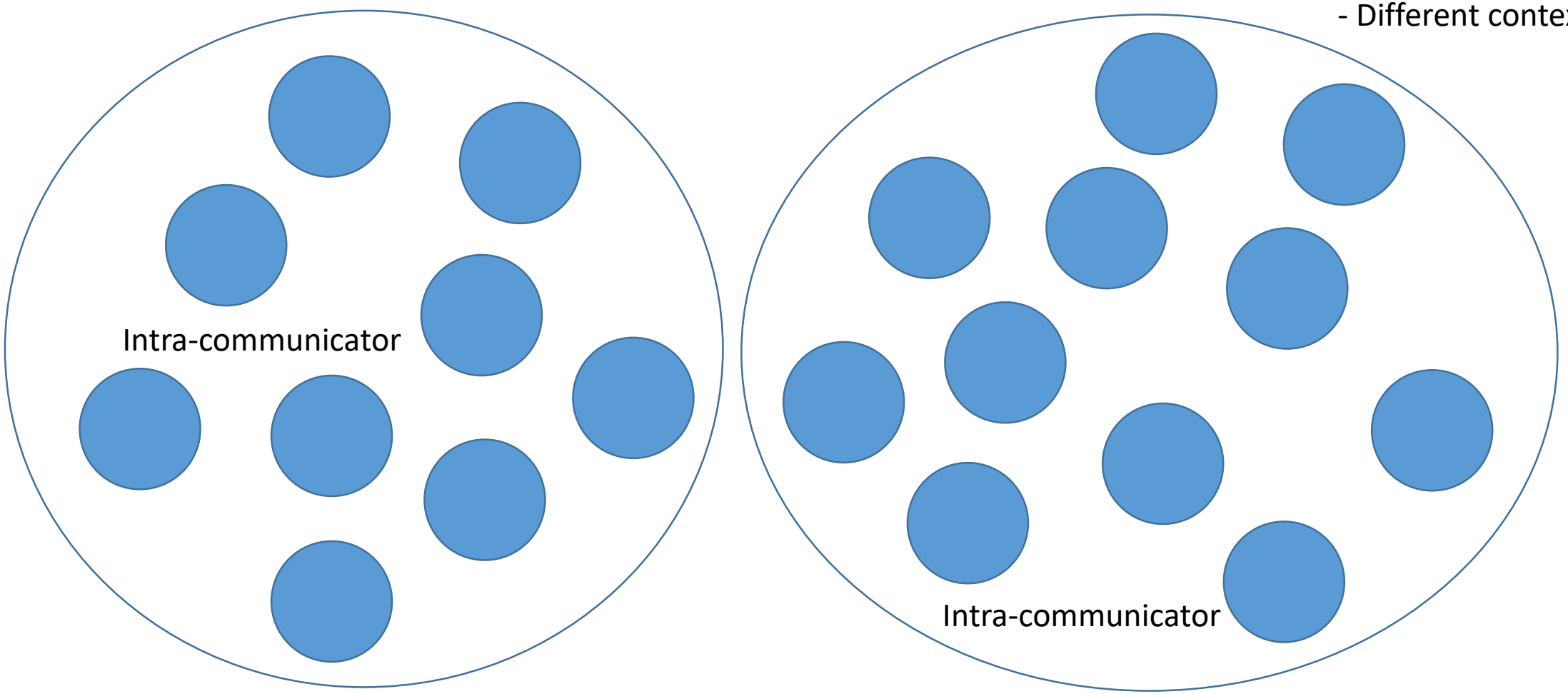
- Ordered set of processes
- Ranks are contiguous
- Base group – associated with MPI_COMM_WORLD
- MPI_Group object
  - MPI_Group_rank, MPI_Group_size
- Unions and intersections of groups
- Not used in communication context

MPI_COMM_WORLD

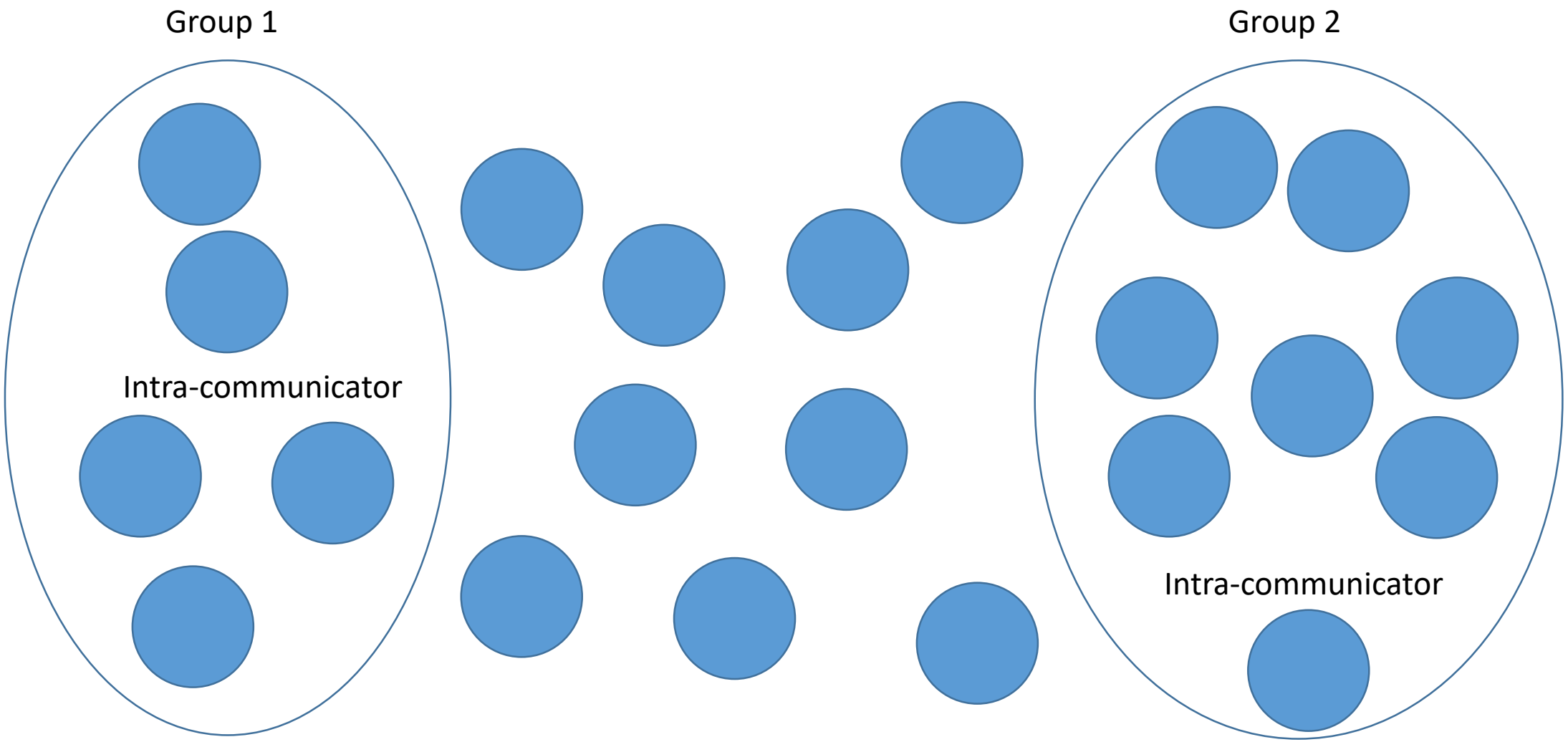


Sub-communicator

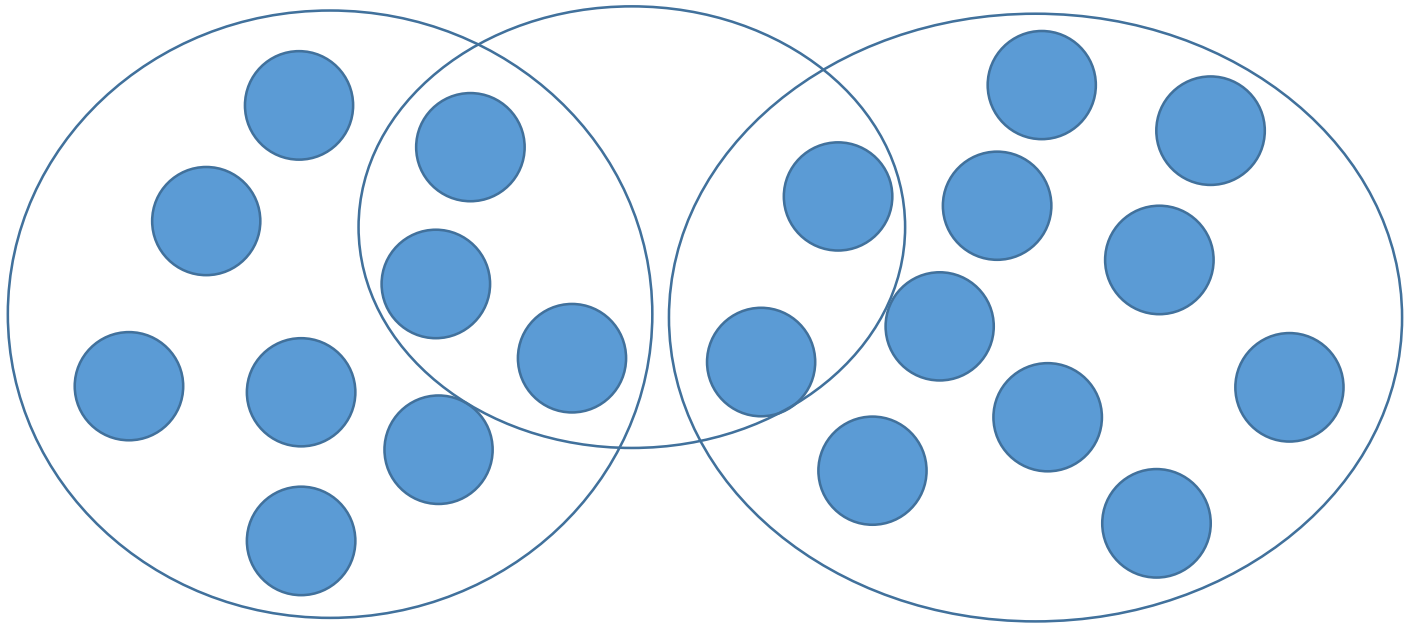
- Logical subset
- Different contexts



Intra-Communicators



Inter-Communicators



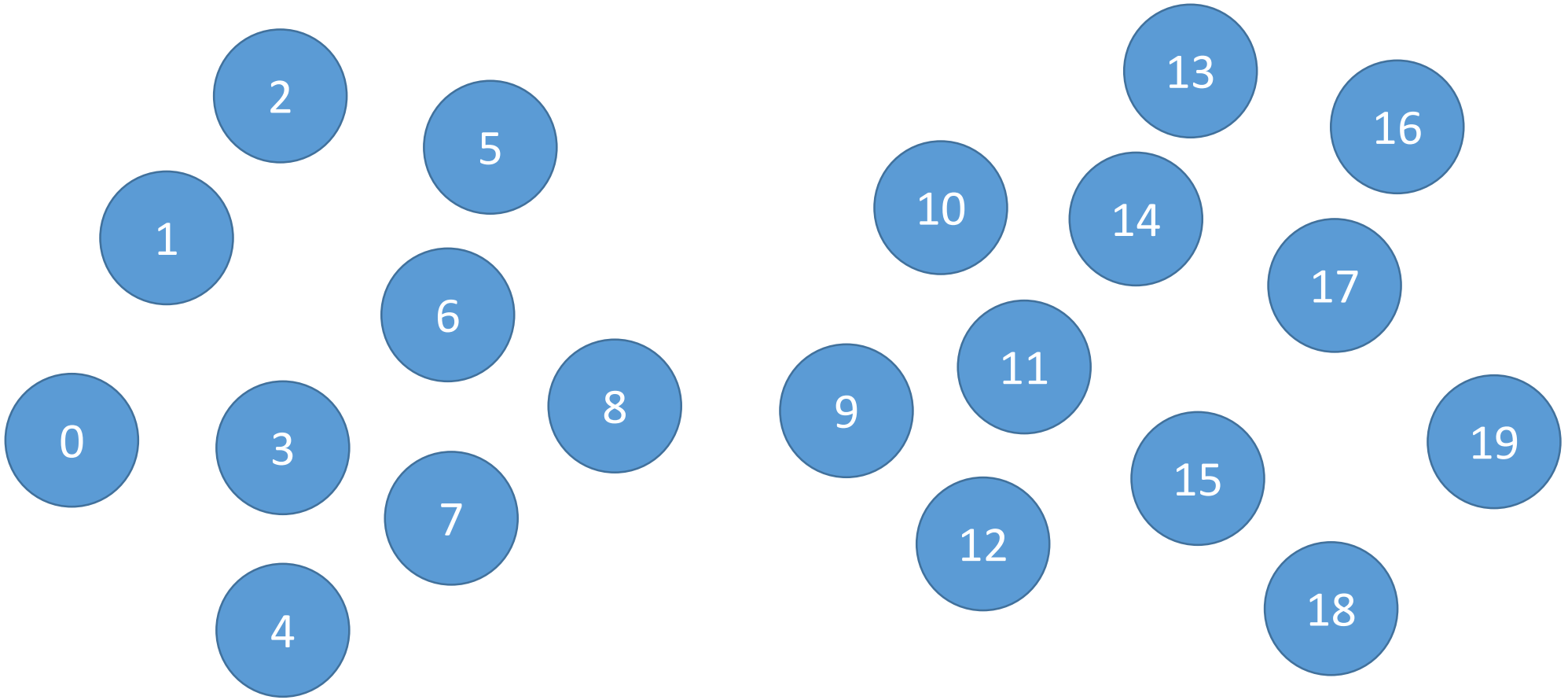
Inter-communicator
between two groups

MPI_COMM_SPLIT

MPI_Comm_split (MPI_Comm oldcomm, int color, int key, MPI_Comm *newcomm)

- Collective call
- Logically divides based on *color*
  - Same color processes form a group
  - Some processes may not be part of newcomm (MPI_UNDEFINED)
- Rank assignment based on *key*

Logical subsets of processes



0 → 0
2 → 1
4 → 2
…

1 → 0
3 → 1
5 → 2
…

How do you assign one color to odd processes and another color to even processes ?
color = rank % 2

Example code

```
int newrank, newsize, color = myrank%2;
MPI_Comm newcomm;

MPI_Comm_split (MPI_COMM_WORLD, color, myrank, &newcomm);

MPI_Comm_rank (newcomm, &newrank);
MPI_Comm_size (newcomm, &newsize);
printf ("%d: %d of %d\n", myrank, newrank, newsize);

MPI_Comm_free (&newcomm);
```

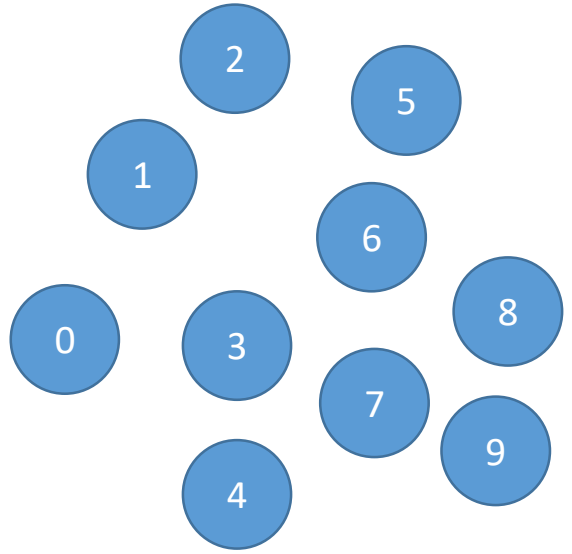
Output for P=8

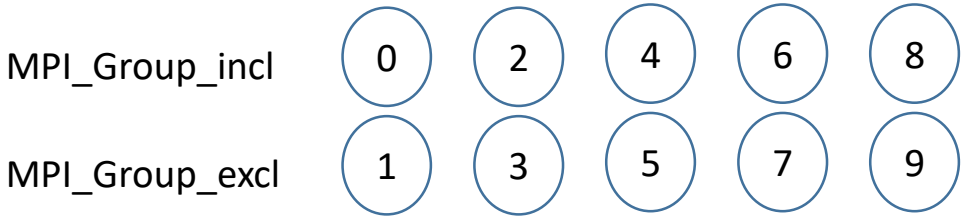
```
0: 0 of 4
1: 0 of 4
2: 1 of 4
3: 1 of 4
4: 2 of 4
5: 2 of 4
6: 3 of 4
7: 3 of 4
```

MPI_Group

```
MPI_Group_incl (
        MPI_Group g_group,
        int N,
        const int ranks[],
        MPI_Group *new_group);
```

```
MPI_Comm_create_group (
        MPI_COMM_WORLD,
        new_group,
        tag,
        &new_comm);
```

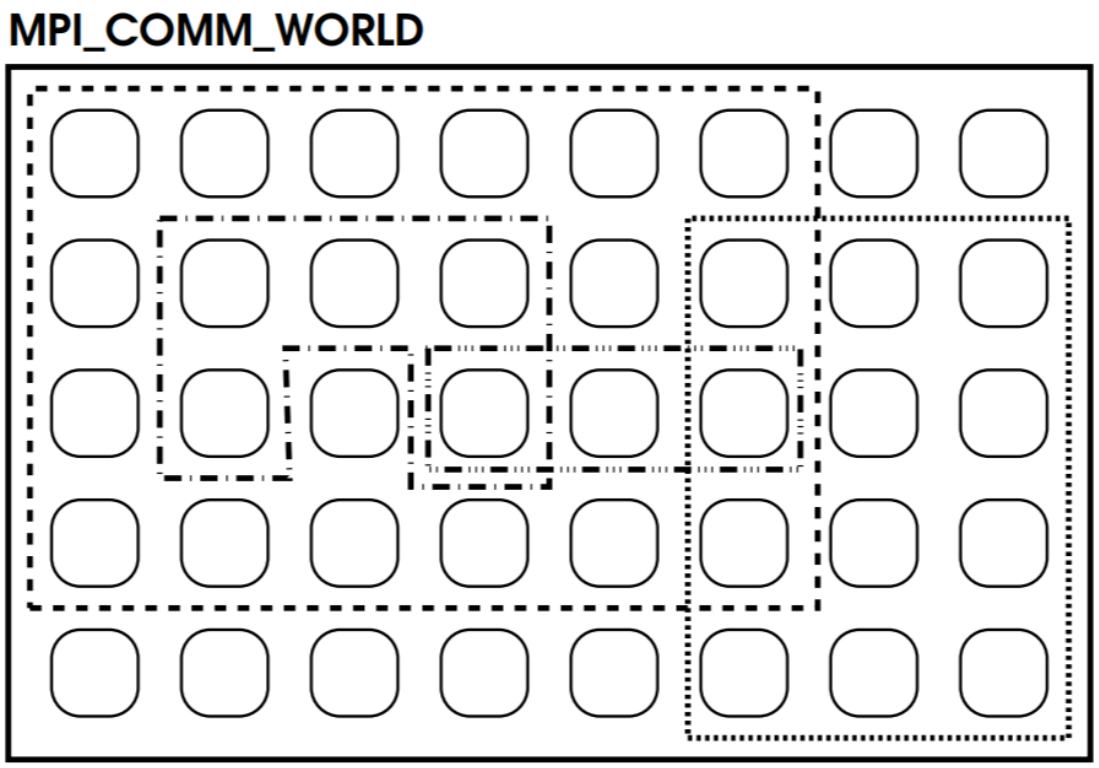




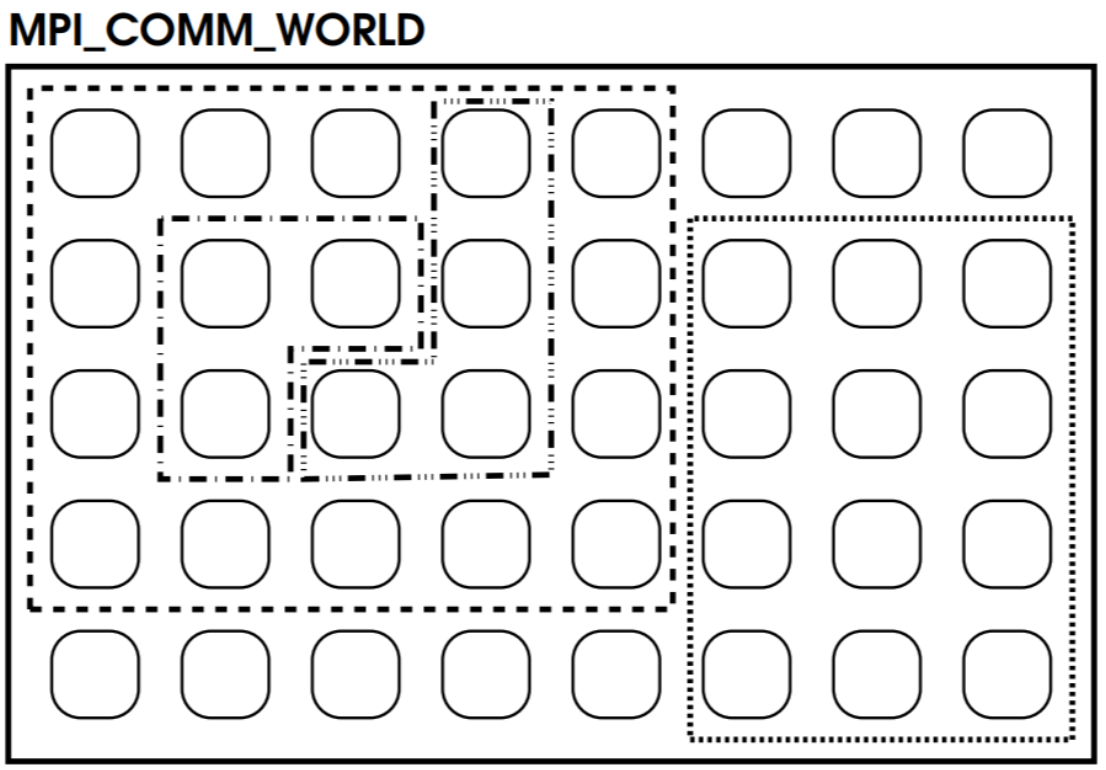
MPI_Group Code

```
int ranks[] = {0,2,4,6,8};

// Construct a group containing even ranks in g_group
MPI_Group new_group;
MPI_Group_incl (g_group, N, ranks, &new_group);

// Create a new communicator based on the group
MPI_Comm new_comm;
MPI_Comm_create_group (MPI_COMM_WORLD, new_group, tag,
&new_comm);
```

MPI Sub-communicators



Overlapping

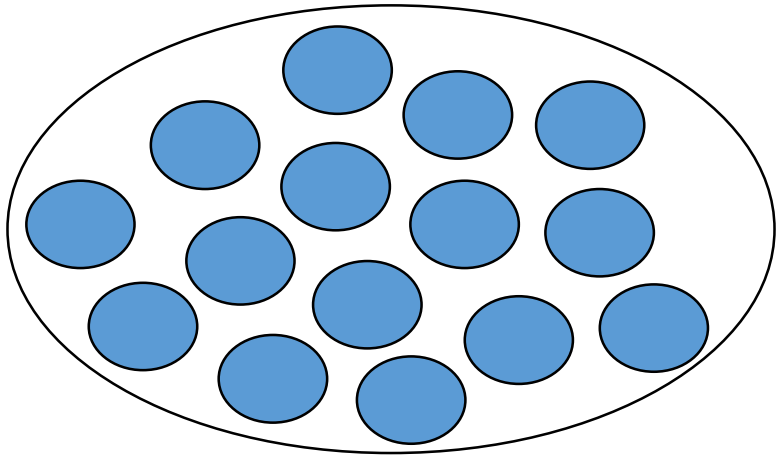


Non-overlapping

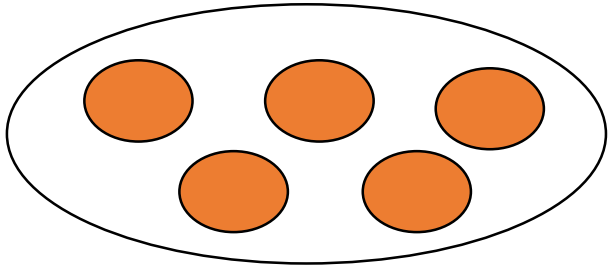
Source: Maclaren slides

Usage of Sub-communicators

- Producer processes



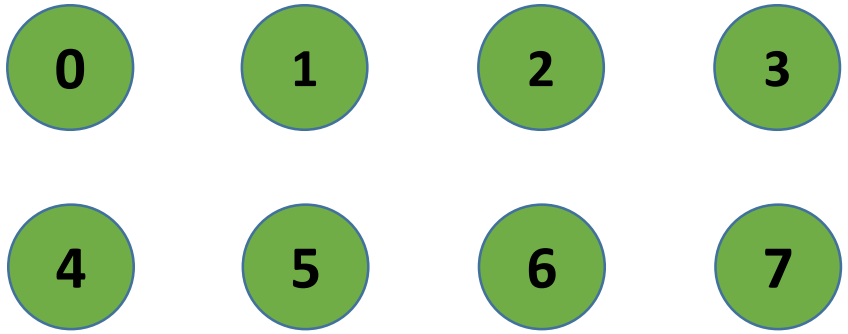
- Consumer processes



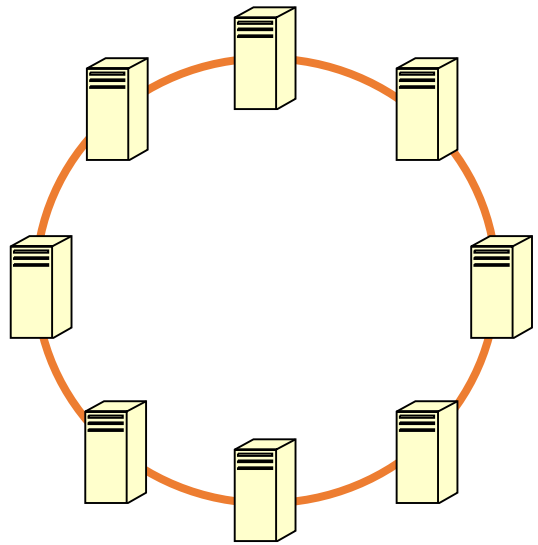
Duplicate Communicator

MPI_Comm_dup (comm, newcomm)

Process Mapping



2 X 4 virtual topology



Physical topology

Groups and Sub-communicators

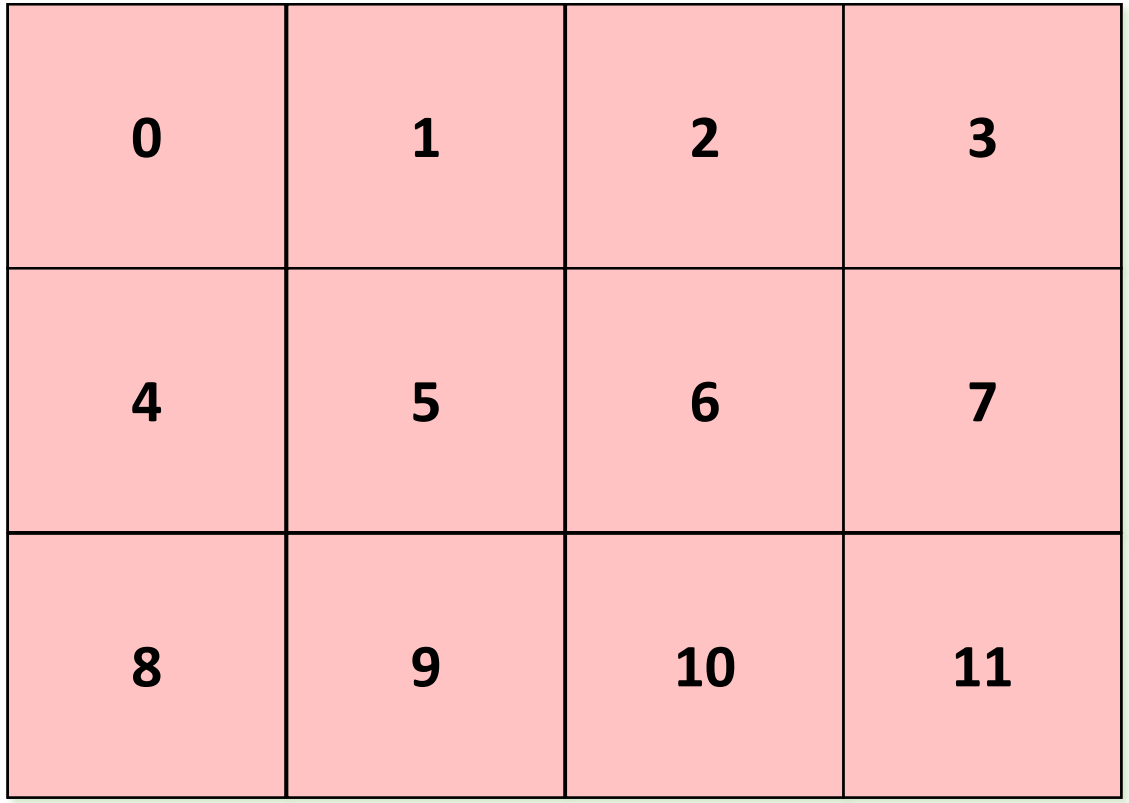
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31

4 x 8 2D virtual process topology

```
// set up rows and columns in 2D virtual topology
int rows = 4;
int cols = numtasks/rows;

// create new group ranks array
int ranks[cols], i, j=-1;
for (i=(myrank/cols)*cols; i<((myrank/cols)+1)*cols; i++)
    ranks[++j] = i;

// get the world group
MPI_Group g_group;
MPI_Comm_group (MPI_COMM_WORLD, &g_group);

// create new groups
MPI_Group new_group;
MPI_Group_incl (g_group, cols, ranks, &new_group);

// create new communicators
MPI_Comm new_comm;
MPI_Comm_create_group (MPI_COMM_WORLD, new_group, myrank/cols, &new_comm);

// size of new communicators
int new_size, new_rank;
MPI_Comm_size (new_comm, &new_size);
MPI_Comm_rank (new_comm, &new_rank);
printf ("Old rank %d, new rank %d\n", myrank, new_rank);
```

Yet another way: int MPI_Group_range_incl (MPI_Group group, int n, int ranges[][3], MPI_Group *newgroup)

MPI_Cart_create



Cartesian topology

MPI_Cart_create (MPI_Comm comm_old, int ndims, int *dims, int *periodic, int reorder, MPI_Comm *comm_cart)

MPI_Cartdim_get
MPI_Cart_rank
MPI_Cart_coords
MPI_Cart_sub

MPI_Cart_create Example

```
ndims = 2
dim[0] = 3 /* rows */, dim[1] = 4 /* columns */
wrap_around[0] = 0, wrap_around[1] = 0
reorder = 0
MPI_Cart_create (MPI_COMM_WORLD, ndims, dims, wrap_around,
reorder, &comm2D)
```

MPI_Cart_shift

int MPI_Cart_shift (MPI_Comm comm, int direction, int disp, int *rank_source, int *rank_dest)

MPI_Cart_create

MPI_Comm_rank

MPI_Cart_coords

MPI_Cart_shift

MPI_Cart_shift Example

```
MPI_Init(&argc, &argv);
MPI_Comm_rank (MPI_COMM_WORLD, &myrank);
MPI_Comm_size (MPI_COMM_WORLD, &size);
MPI_Comm comm2D;

dim[0] = 3 /* rows */, dim[1] = 4 /* columns */;
wrap_around[0] = 0, wrap_around[1] = 0;
reorder = 0;

MPI_Cart_create (MPI_COMM_WORLD, ndims, dim, wrap_around, reorder, &comm2D);

MPI_Comm_rank (comm2D, &newrank);
MPI_Comm_size (comm2D, &newsize);

MPI_Cart_shift (comm2D, 0, 1, &source, &dest);
printf ("Rank %d, new rank %d, source %d dest %d\n", myrank, newrank, source, dest);

MPI_Comm_free (&comm2D);
```

```
class $ mpirun -np 12 ./cart | sort -k2n
Rank 0, new rank 0, source -1 dest 4
Rank 1, new rank 1, source -1 dest 5
Rank 2, new rank 2, source -1 dest 6
Rank 3, new rank 3, source -1 dest 7
Rank 4, new rank 4, source 0 dest 8
Rank 5, new rank 5, source 1 dest 9
Rank 6, new rank 6, source 2 dest 10
Rank 7, new rank 7, source 3 dest 11
Rank 8, new rank 8, source 4 dest -1
Rank 9, new rank 9, source 5 dest -1
Rank 10, new rank 10, source 6 dest -1
Rank 11, new rank 11, source 7 dest -1
```

0	1	2	3
4	5	6	7
8	9	10	11

```
class $ mpirun -np 12 ./cart | sort -k2n
Rank 0, new rank 0, source 4 dest -1
Rank 1, new rank 1, source 5 dest -1
Rank 2, new rank 2, source 6 dest -1
Rank 3, new rank 3, source 7 dest -1
Rank 4, new rank 4, source 8 dest 0
Rank 5, new rank 5, source 9 dest 1
Rank 6, new rank 6, source 10 dest 2
Rank 7, new rank 7, source 11 dest 3
Rank 8, new rank 8, source -1 dest 4
Rank 9, new rank 9, source -1 dest 5
Rank 10, new rank 10, source -1 dest 6
Rank 11, new rank 11, source -1 dest 7
```

MPI_Cart_sub

```
ndims = 2

dim[0] = 3, dim[1] = 3

wrap_around[0] = 0, wrap_around[1] = 0

reorder = 0

MPI_Cart_create (MPI_COMM_WORLD, ndims, dims, wrap_around,
reorder, &comm2D)


remains[0] = 0, remains[1] = 1   //column dimension

MPI_Cart_sub (comm2D, remains, &comm1D_row)

remains[0] = 1, remains[1] = 0

MPI_Cart_sub (comm2D, remains, &comm1D_col)


MPI_Reduce (&val, &rowmax, 1, MPI_INT, MPI_MAX, 0, comm1D_row);

MPI_Reduce (&rowmax, &max, 1, MPI_INT, MPI_MAX, 0, comm1D_col);
```

0 (0,0) **val=78**	**1 (0,1)** **72**	**2 (0,2)** **70**
3 (1,0) **81**	**4 (1,1)** **87**	**5 (1,2)** **80**
6 (2,0) **77**	**7 (2,1)** **78**	**8 (2,2)** **75**