

All communication described thus far has involved communication between processes that are members of the same group. This type of communication is called “intra-communication” and the communicator used is called an “intra-communicator,” as we have noted earlier in the chapter.

In modular and multi-disciplinary applications, different process groups execute distinct modules and processes within different modules communicate with one another in a pipeline or a more general module graph. In these applications, the most natural way for a process to specify a target process is by the rank of the target process within the target group. In applications that contain internal user-level servers, each server may be a process group that provides services to one or more clients, and each client may be a process group that uses the services of one or more servers. It is again most natural to specify the target process by rank within the target group in these applications. This type of communication is called “inter-communication” and the communicator used is called an “inter-communicator,” as introduced earlier.

An inter-communication is a point-to-point communication between processes in different groups. The group containing a process that initiates an inter-communication operation is called the “local group,” that is, the sender in a send and the receiver in a receive. The group containing the target process is called the “remote group,” that is, the receiver in a send and the sender in a receive. As in intra-communication, the target process is specified using a (communicator, rank) pair. Unlike intra-communication, the rank is relative to a second, remote group.

*Advice to implementors.* To be able to use the predefined C functions `MPI_COMM_NULL_COPY_FN` or `MPI_COMM_DUP_FN` as `comm_copy_attr_fn` argument and/or `MPI_COMM_NULL_DELETE_FN` as the `comm_delete_attr_fn` argument in a call to the C++ routine `MPI::Comm::Create_keyval`, this routine may be overloaded with 3 additional routines that accept the C functions as the first, the second, or both input arguments (instead of an argument that matches the C++ prototype). (*End of advice to implementors.*)

*Advice to users.* If a user wants to write a “wrapper” routine that internally calls `MPI::Comm::Create_keyval` and `comm_copy_attr_fn` `comm_delete_attr_fn` are arguments of this wrapper routine, and if this wrapper routine should be callable with both user-defined C++ copy and delete functions and with the predefined C functions, then the same overloading as described above in the advice to implementors may be necessary. (*End of advice to users.*)

All inter-communicator constructors are blocking and require that the local and remote groups be disjoint.

*Advice to users.* The groups must be disjoint for several reasons. Primarily, this is the intent of the intercommunicators — to provide a communicator for communication between disjoint groups. This is reflected in the definition of `MPI_INTERCOMM_MERGE`, which allows the user to control the ranking of the processes in the created intracommunicator; this ranking makes little sense if the groups are not disjoint. In addition, the natural extension of collective operations to inter-communicators makes the most sense when the groups are disjoint. (*End of advice to users.*)

Here is a summary of the properties of inter-communication and inter-communicators:

ticket107.