7.5.4 Distributed (Graph) Constructor

The general graph constructor assumes that each process passes the full (global) communication graph to the call. This limits the scalability of this constructor. With the distributed graph interface, the communication graph is specified in a fully distributed fashion. Each process specifies only the part of the communication graph of which it is aware. Typically, this could be the set of processes from which the process will eventually receive or get data, or the set of processes to which the process will send or put data, or some combination of such edges. Two different interfaces can be used to create a distributed graph topology. MPI_DIST_GRAPH_CREATE_ADJACENT creates a distributed graph communicator with each process specifying all of its incoming and outgoing (adjacent) edges in the logical communication graph and thus requires minimal communication during creation. MPI_DIST_GRAPH_CREATE provides full flexibility, and processes can indicate that communication will occur between other pairs of processes.

To provide better possibilities for optimization by the MPI library, the distributed graph constructors permit weighted communication edges and take an info argument that can further influence process reordering or other optimizations performed by the MPI library. For example, hints can be provided on how edge weights are to be interpreted, the quality of the reordering, and/or the time permitted for the MPI library to process the graph.

MPI_DIST_GRAPH_CREATE_ADJACENT(comm_old, indegree, sources, sourceweights, outdegree, destinations, destweights, info, reorder, comm_dist_graph)

IN	comm_old	input communicator (handle)
IN	indegree	size of sources and sourceweights arrays (non-negative integer)
IN	sources	ranks of processes for which the calling process is a destination (array of non-negative integers)
IN	sourceweights	weights of the edges into the calling process (array of non-negative integers)
IN	outdegree	size of destinations and destweights arrays (non-negative integer)
IN	destinations	ranks of processes for which the calling process is a source (array of non-negative integers)
IN	destweights	weights of the edges out of the calling process (array of non-negative integers)
IN	info	hints on optimization and interpretation of weights (handle)
IN	reorder	the ranks may be reordered (true) or not (false) (logical)
OUT	comm_dist_graph	communicator with distributed graph topology (handle)

```
int MPI_Dist_graph_create_adjacent(MPI_Comm comm_old, int indegree,
             int sources[], int sourceweights[], int outdegree,
```

```
              int destinations[], int destweights[], MPI_Info info,
              int reorder, MPI_Comm *comm_dist_graph)

MPI_DIST_GRAPH_CREATE_ADJACENT(COMM_OLD, INDEGREE, SOURCES, SOURCEWEIGHTS,
              OUTDEGREE, DESTINATIONS, DESTWEIGHTS, INFO, REORDER,
              COMM_DIST_GRAPH, IERROR)
    INTEGER COMM_OLD, INDEGREE, SOURCES(*), SOURCEWEIGHTS(*), OUTDEGREE,
    DESTINATIONS(*), DESTWEIGHTS(*), INFO, COMM_DIST_GRAPH, IERROR
    LOGICAL REORDER

{MPI::Distgraphcomm MPI::Intracomm::Dist_graph_create_adjacent(int
              indegree, const int sources[], const int sourceweights[],
              int outdegree, const int destinations[],
              const int destweights[], const MPI::Info& info, bool reorder)
              const (binding deprecated, see Section 15.2) } ticket150.

{MPI::Distgraphcomm                                                ticket150.
              MPI::Intracomm::Dist_graph_create_adjacent(int indegree,
              const int sources[], int outdegree, const int destinations[],
              const MPI::Info& info, bool reorder) const (binding deprecated,
              see Section 15.2) }                                  ticket150.
```

MPI_DIST_GRAPH_CREATE_ADJACENT returns a handle to a new communicator to which the distributed graph topology information is attached. Each process passes all information about the edges to its neighbors in the virtual distributed graph topology. The calling processes must ensure that each edge of the graph is described in the source and in the destination process with the same weights. If there are multiple edges for a given (source,dest) pair, then the sequence of the weights of these edges does not matter. The complete communication topology is the combination of all edges shown in the sources arrays of all processes in comm_old, which must be identical to the combination of all edges shown in the destinations arrays. Source and destination ranks must be process ranks of comm_old. This allows a fully distributed specification of the communication graph. Isolated processes (i.e., processes with no outgoing or incoming edges, that is, processes that have specified indegree and outdegree as zero and that thus do not occur as source or destination rank in the graph specification) are allowed.

The call creates a new communicator comm_dist_graph of distributed graph topology type to which topology information has been attached. The number of processes comm_dist_graph is identical to the number of processes in comm_old. The call to MPI_DIST_GRAPH_CREATE_ADJACENT is collective.

Weights are specified as non-negative integers and can be used to influence the process remapping strategy and other internal MPI optimizations. For instance, approximate count arguments of later communication calls along specific edges could be used as their edge weights. Multiplicity of edges can likewise indicate more intense communication between pairs of processes. However, the exact meaning of edge weights is not specified by the MPI standard and is left to the implementation. In C or Fortran, an application can supply the special value MPI_UNWEIGHTED for the weight array to indicate that all edges have the same (effectively no) weight. In C++, this constant does not exist and the weights argument may be omitted from the argument list. It is erroneous to supply MPI_UNWEIGHTED, or in C++ omit the weight arrays, for some but not all processes of comm_old. Note that

MPI_UNWEIGHTED is not a special weight value; rather it is a special value for the total array argument. In C, one would expect it to be NULL. In Fortran, MPI_UNWEIGHTED is an object like MPI_BOTTOM (not usable for initialization or assignment). See Section 2.5.4

The meaning of the info and reorder arguments is defined in the description of the following routine.

MPI_DIST_GRAPH_CREATE(comm_old, n, sources, degrees, destinations, weights, info, reorder, comm_dist_graph)

IN	comm_old	input communicator (handle)
IN	n	number of source nodes for which this process specifies edges (non-negative integer)
IN	sources	array containing the n source nodes for which this process specifies edges (array of non-negative integers)
IN	degrees	array specifying the number of destinations for each source node in the source node array (array of non-negative integers)
IN	destinations	destination nodes for the source nodes in the source node array (array of non-negative integers)
IN	weights	weights for source to destination edges (array of non-negative integers)
IN	info	hints on optimization and interpretation of weights (handle)
IN	reorder	the process may be reordered (true) or not (false) (logical)
OUT	comm_dist_graph	communicator with distributed graph topology added (handle)

```
int MPI_Dist_graph_create(MPI_Comm comm_old, int n, int sources[],
              int degrees[], int destinations[], int weights[],
              MPI_Info info, int reorder, MPI_Comm *comm_dist_graph)
```

```
MPI_DIST_GRAPH_CREATE(COMM_OLD, N, SOURCES, DEGREES, DESTINATIONS, WEIGHTS,
              INFO, REORDER, COMM_DIST_GRAPH, IERROR)
    INTEGER COMM_OLD, N, SOURCES(*), DEGREES(*), DESTINATIONS(*),
    WEIGHTS(*), INFO, COMM_DIST_GRAPH, IERROR
    LOGICAL REORDER
```

ticket150.

```
{MPI::Distgraphcomm MPI::Intracomm::Dist_graph_create(int n,
              const int sources[], const int degrees[], const int
              destinations[], const int weights[], const MPI::Info& info,
              bool reorder) const (binding deprecated, see Section 15.2) }
```

ticket150.

ticket150.

```
{MPI::Distgraphcomm MPI::Intracomm::Dist_graph_create(int n,
              const int sources[], const int degrees[],
              const int destinations[], const MPI::Info& info, bool reorder)
              const (binding deprecated, see Section 15.2) }
```

ticket150.