If comm is an intracommunicator, the outcome of a call to MPI_ALLGATHER(...) is as if all processes executed n calls to

```
MPI_G[ticket118.][ATHER]ather(sendbuf,sendcount,sendtype,recvbuf,recvcount,
                                        recvtype,root,comm)[ticket120.][,]
```

for root = 0 , ..., n-1. The rules for correct usage of MPI_ALLGATHER are easily found from the corresponding rules for MPI_GATHER.

The "in place" option for intracommunicators is specified by passing the value MPI_IN_PLACE to the argument sendbuf at all processes. sendcount and sendtype are ignored. Then the input data of each process is assumed to be in the area where that process would receive its own contribution to the receive buffer.

If comm is an intercommunicator, then each process [in group A contributes a data item; these items]of one group (group A) contributes sendcount data items; these data are concatenated and the result is stored at each process in [group B]the other group (group B). Conversely the concatenation of the contributions of the processes in group B is stored at each process in group A. The send buffer arguments in group A must be consistent with the receive buffer arguments in group B, and vice versa. ticket132. ticket132.

> *Advice to users.* The communication pattern of MPI_ALLGATHER executed on an intercommunication domain need not be symmetric. The number of items sent by processes in group A (as specified by the arguments sendcount, sendtype in group A and the arguments recvcount, recvtype in group B), need not equal the number of items sent by processes in group B (as specified by the arguments sendcount, sendtype in group B and the arguments recvcount, recvtype in group A). In particular, one can move data in only one direction by specifying sendcount = 0 for the communication in the reverse direction.
>
> (*End of advice to users.*)

MPI_ALLGATHERV(sendbuf, sendcount, sendtype, recvbuf, recvcounts, displs, recvtype, comm)

IN	sendbuf	starting address of send buffer (choice)
IN	sendcount	number of elements in send buffer (non-negative integer)
IN	sendtype	data type of send buffer elements (handle)
OUT	recvbuf	address of receive buffer (choice)
IN	recvcounts	non-negative integer array (of length group size) containing the number of elements that are received from each process
IN	displs	integer array (of length group size). Entry `i` specifies the displacement (relative to recvbuf) at which to place the incoming data from process `i`
IN	recvtype	data type of receive buffer elements (handle)
IN	comm	communicator (handle)

```
int MPI_Allgatherv(void* sendbuf, int sendcount, MPI_Datatype sendtype,
              void* recvbuf, int *recvcounts, int *displs,
              MPI_Datatype recvtype, MPI_Comm comm)
```

```
MPI_ALLGATHERV(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNTS, DISPLS,
              RECVTYPE, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNTS(*), DISPLS(*), RECVTYPE, COMM,
    IERROR
```

ticket150.

```
{void MPI::Comm::Allgatherv(const void* sendbuf, int sendcount, const
              MPI::Datatype& sendtype, void* recvbuf,
              const int recvcounts[], const int displs[],
              const MPI::Datatype& recvtype) const = 0 (binding deprecated, see
              Section 15.2) }
```

MPI_ALLGATHERV can be thought of as MPI_GATHERV, but where all processes receive the result, instead of just the root. The block of data sent from the `j`-th process is received by every process and placed in the `j`-th block of the buffer recvbuf. These blocks need not all be the same size.

The type signature associated with sendcount, sendtype, at process `j` must be equal to the type signature associated with recvcounts[j], recvtype at any other process.

If comm is an intracommunicator, the outcome is as if all processes executed calls to

```
MPI_GATHERV(sendbuf,sendcount,sendtype,recvbuf,recvcounts,displs,
                                                    recvtype,root,comm),
```

for `root = 0 , ..., n-1`. The rules for correct usage of MPI_ALLGATHERV are easily found from the corresponding rules for MPI_GATHERV.

The "in place" option for intracommunicators is specified by passing the value

ticket120. MPI_IN_PLACE to the argument sendbuf at all processes. In such a case, sendcount and

sendtype are ignored, and the input data of each process is assumed to be in the area where that process would receive its own contribution to the receive buffer.

If comm is an intercommunicator, then each process [in group A contributes a data item; these items]of one group (group A) contributes sendcount data items; these data are concatenated and the result is stored at each process in [group B]the other group (group B). Conversely the concatenation of the contributions of the processes in group B is stored at each process in group A. The send buffer arguments in group A must be consistent with the receive buffer arguments in group B, and vice versa.

5.7.1 Example[s] using MPI_ALLGATHER[, MPI_ALLGATHERV]

The example[s] in this section use intracommunicators.

Example 5.14 The all-gather version of Example 5.2. Using MPI_ALLGATHER, we will gather 100 ints from every process in the group to every process.

```
MPI_Comm comm;
int gsize,sendarray[100];
int *rbuf;
...
MPI_Comm_size( comm, &gsize);
rbuf = (int *)malloc(gsize*100*sizeof(int));
MPI_Allgather( sendarray, 100, MPI_INT, rbuf, 100, MPI_INT, comm);
```

After the call, every process has the group-wide concatenation of the sets of data.

5.8 All-to-All Scatter/Gather

MPI_ALLTOALL(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype, comm)

IN	sendbuf	starting address of send buffer (choice)
IN	sendcount	number of elements sent to each process (non-negative integer)
IN	sendtype	data type of send buffer elements (handle)
OUT	recvbuf	address of receive buffer (choice)
IN	recvcount	number of elements received from any process (non-negative integer)
IN	recvtype	data type of receive buffer elements (handle)
IN	comm	communicator (handle)

```
int MPI_Alltoall(void* sendbuf, int sendcount, MPI_Datatype sendtype,
             void* recvbuf, int recvcount, MPI_Datatype recvtype,
             MPI_Comm comm)
```

```
MPI_ALLTOALL(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT, RECVTYPE,
             COMM, IERROR)
```