



MPI in Small Bites

PPCES 2026

HPC.NRW Competence Network



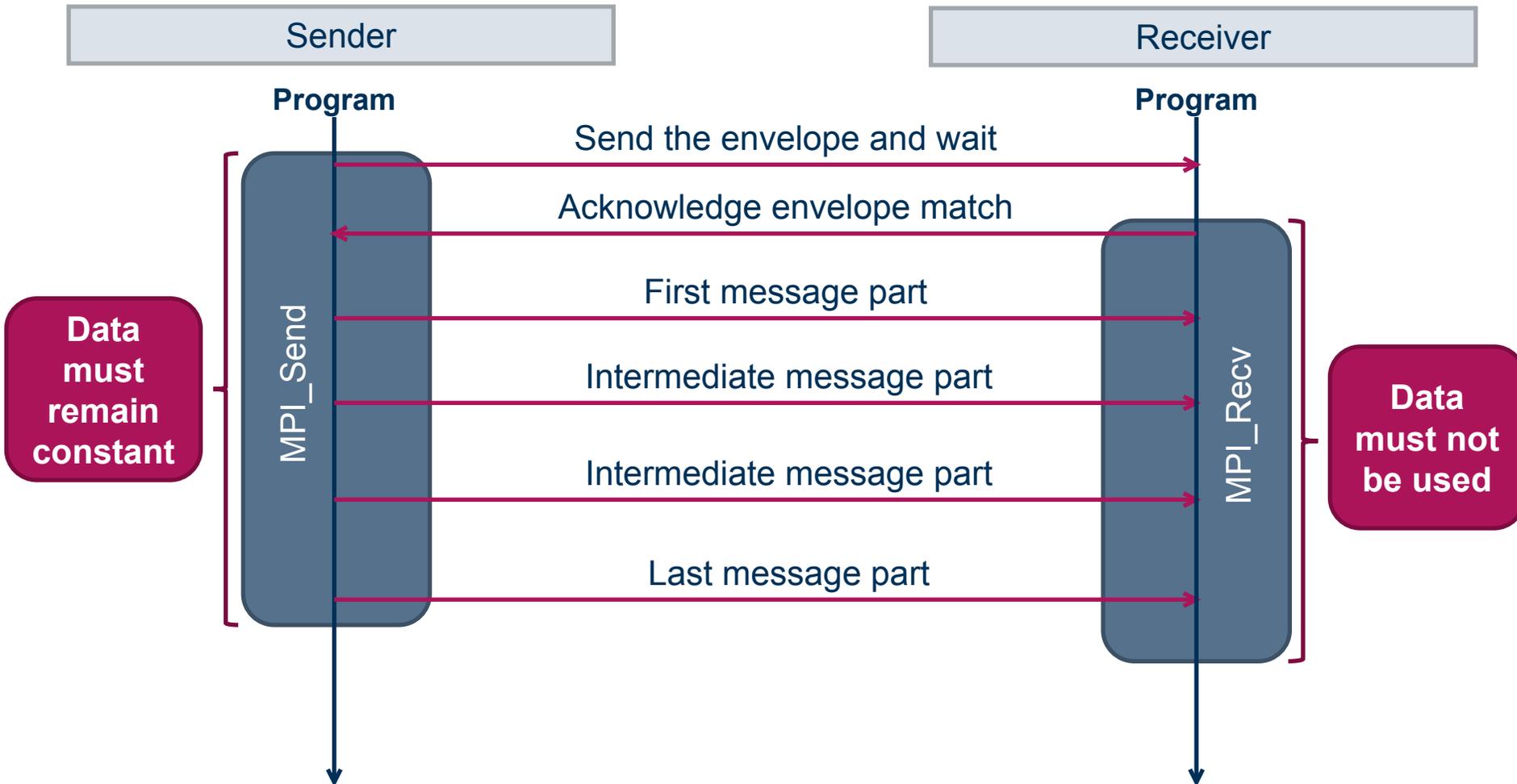
THE COMPETENCE NETWORK FOR HIGH PERFORMANCE COMPUTING IN NRW.

Non-blocking Point-to-Point Communication

HPC.NRW Competence Network

MPI in Small Bites

Blocking send (w/o buffering) and receive calls:



- Return **before** associated operation is complete
- Separate call needed to complete operation
- Track non-blocking operation using a **request handle**:
 - C: **MPI_Request**
 - Fortran: **INTEGER**
 - Fortran 2008: **TYPE(MPI_Request)**
- Operation progress is implementation dependent
 - Within MPI functions (e.g., communication functions or other ‘expensive’ function calls)
 - Progress thread
 - Hardware support
- Used to overlap communication and computation and **to prevent possible deadlocks**

- Non-blocking procedures often have an 'I' (capital i) prefix
 - Note: not all non-blocking functions have this prefix, but are non-blocking nonetheless
- Initiation of non-blocking send and receive operations:

```
MPI_Isend (void *data, int count, MPI_Datatype dataType,  
          int dest, int tag, MPI_Comm comm, MPI_Request *request)
```

```
MPI_Irecv (void *data, int count, MPI_Datatype dataType,  
          int source, int tag, MPI_Comm comm, MPI_Request *request)
```

- **request:** on success set to the handle of the non-blocking operation

- Blocking for completion
- Returning status objects (as a blocking receive would)
 - Use **MPI_STATUS_IGNORE** to omit return of status object
- Waiting for a single request to complete:

```
MPI_Wait (MPI_Request *request, MPI_Status *status)
```

- Waiting for any single request out of multiple to complete:

```
MPI_Waitany (int count, MPI_Request array_of_requests[],  
             int* index, MPI_Status *status)
```

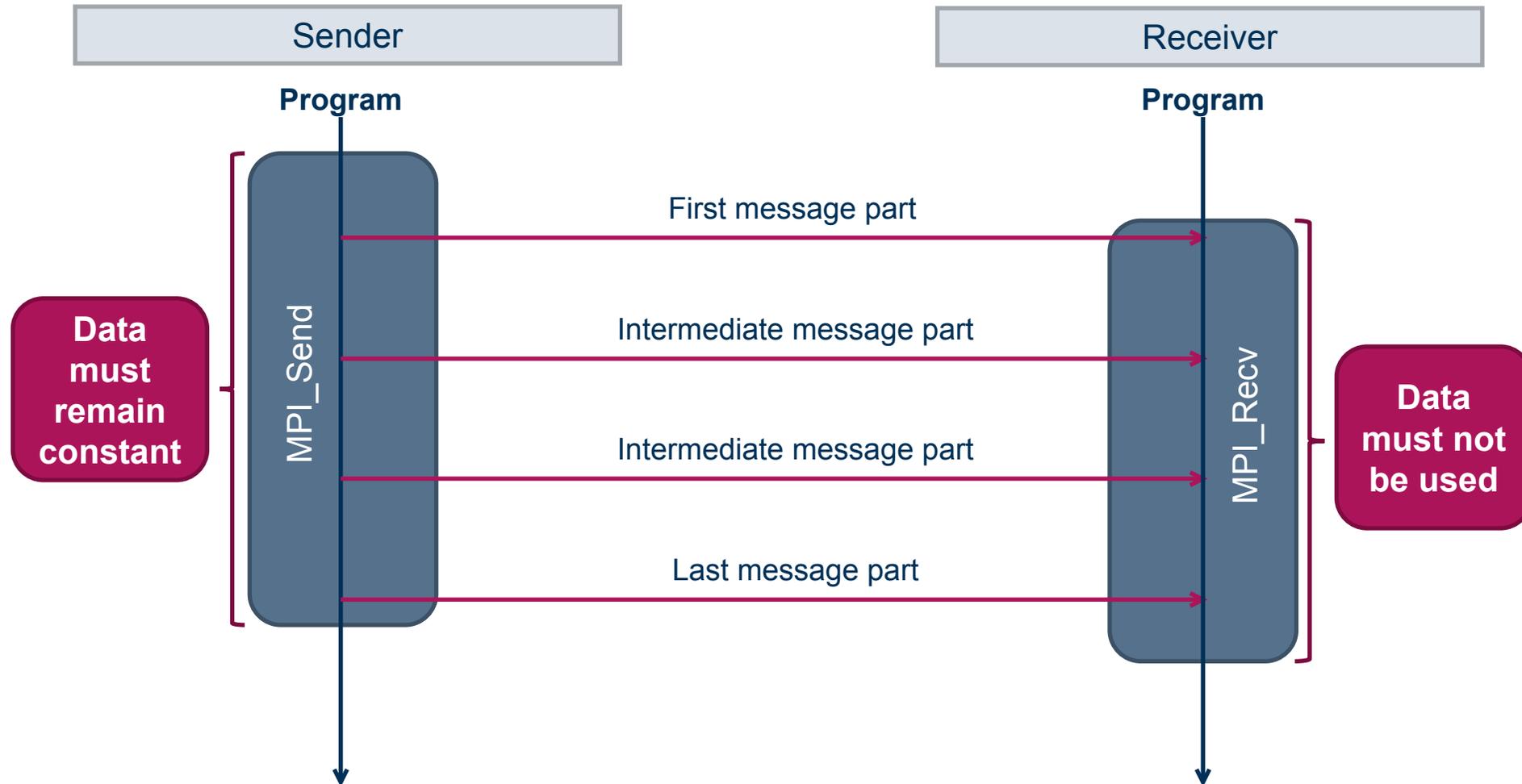
- Waiting for a multiple (not necessarily all) requests out of multiple to complete:

```
MPI_Waitsome (int incount, MPI_Request array_of_requests[],  
             int* outcount, int array_of_indices[],  
             MPI_Status array_of_status[])
```

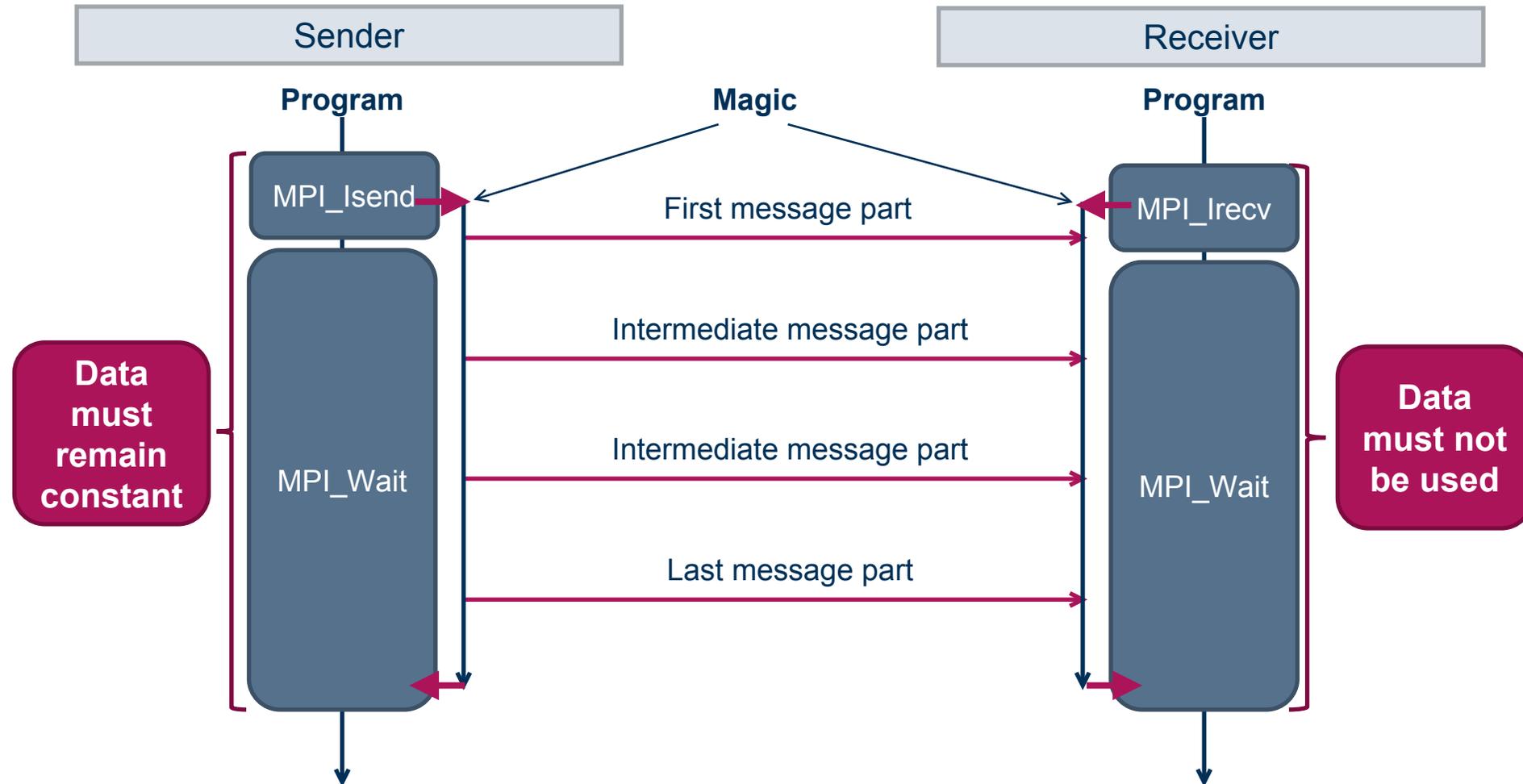
- Use **MPI_STATUSES_IGNORE** to omit return of status objects
 - Returns with outcount set to **MPI_UNDEFINED** on no active requests
- Waiting for all requests out of multiple to complete:

```
MPI_Waitall (int count, MPI_Request array_of_requests[],  
            MPI_Status array_of_statuses[])
```

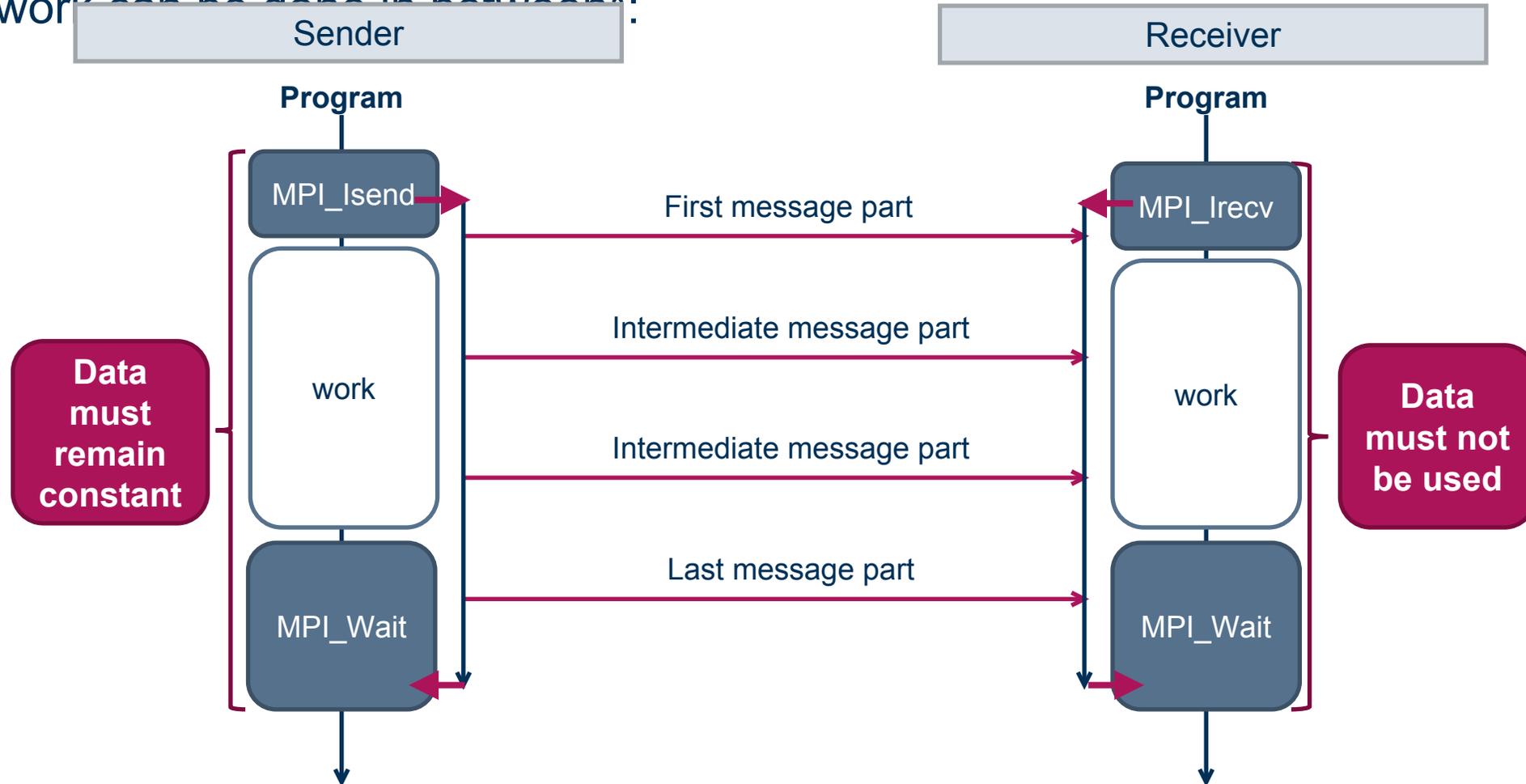
Communication-Computation Overlap



Communication-Computation Overlap

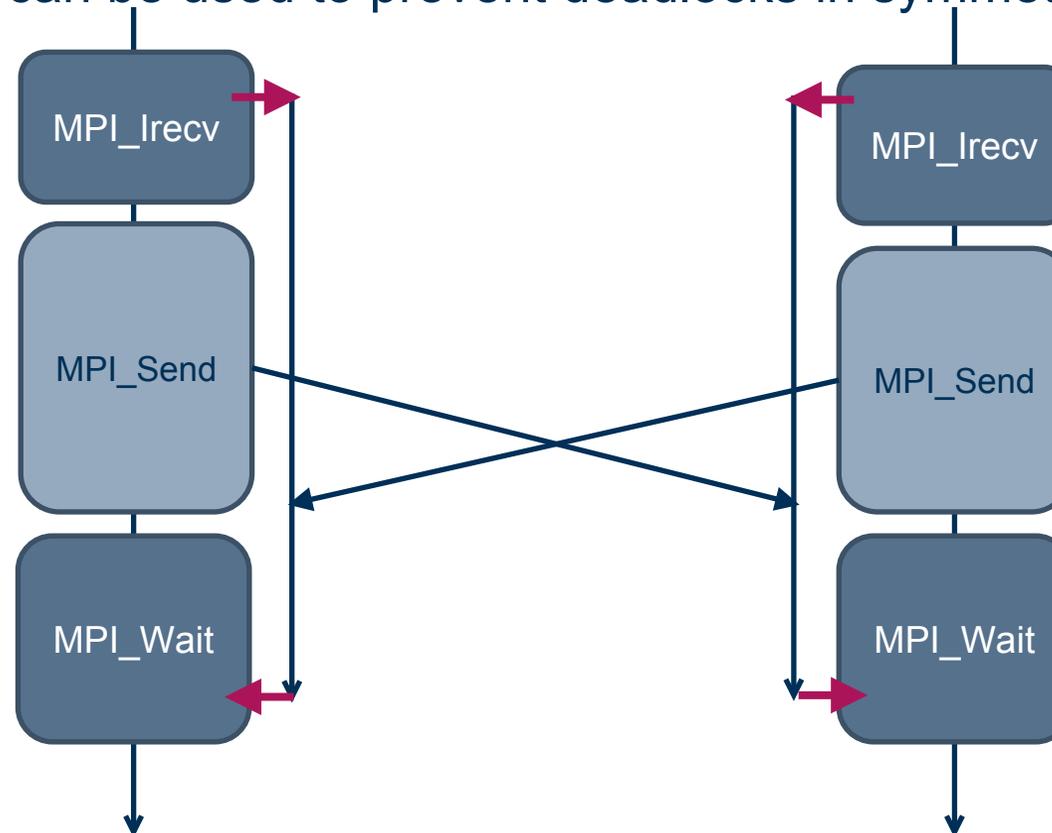


- Other work can be done in between*:



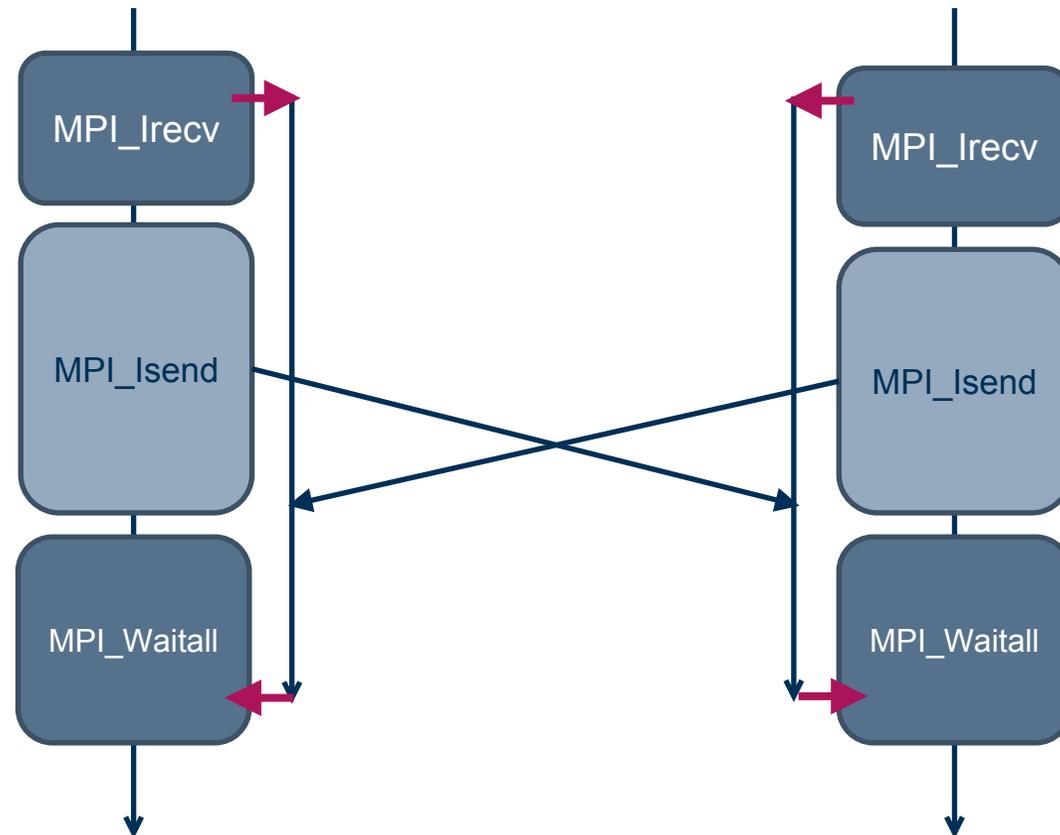
Deadlock Prevention (and Communication-Communication Overlap)

- Non-blocking operations can be used to prevent deadlocks in symmetric code:



- That is how MPI_Sendrecv can be implemented

Deadlock Prevention (and Communication-Communication Overlap)



- Does **NOT** block for completion
 - Indicates whether completion occurred
- Returning status objects (as a blocking receive would)
 - Use **MPI_STATUS_IGNORE** to omit return of status object
- Test for completion of a single request:

```
MPI_Test (MPI_Request *request, int* flag, MPI_Status *status)
```

- Test for completion of any single request out of multiple:

```
MPI_Testany (int count, MPI_Request array_of_requests[],  
             int* index, int* flag, MPI_Status *status)
```

- Test for completion of multiple (not necessarily all) requests out of multiple:

```
MPI_Testsome (int incount, MPI_Request array_of_requests[],  
              int* outcount, int array_of_indices[],  
              MPI_Status array_of_status[])
```

- Flag is implicit in **outcount**
 - Use **MPI_STATUSES_IGNORE** to omit return of status objects
- Test for completion of all requests out of multiple:

```
MPI_Testall (int count, MPI_Request array_of_requests[], int* flag,  
             MPI_Status array_of_statuses[])
```

- Test or wait for **local** completion
- Successful requests are set to `MPI_REQUEST_NULL` on completion

- If called with a null request (`MPI_REQUEST_NULL`):
 - **MPI_Wait** returns immediately with an empty **status**
 - **MPI_Test** sets **flag** to **true** and returns an empty **status**
 - Ignored in the presence of other valid requests