```
MPI_Comm_rank( MPI_COMM_WORLD, &rank);
if (rank==0) {
    MPI_Send( send_data1, n, MPI_INT, 2, 0, MPI_COMM_WORLD);
    MPI_Send( send_data2, n, MPI_INT, 1, 0, MPI_COMM_WORLD);
}
else if (rank==1) {
    MPI_Recv( recv_data1, n, MPI_INT, 0, 0, MPI_COMM_WORLD,
        MPI_STATUS_IGNORE);
    MPI_Send( recv_data1, n, MPI_INT, 2, 0, MPI_COMM_WORLD);
}
else if (rank==2) {
    MPI_Recv( recv_data1, n, MPI_INT, MPI_ANY_SOURCE, 0,
        MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    MPI_Recv( recv_data2, n, MPI_INT, MPI_ANY_SOURCE, 0,
        MPI_COMM_WORLD, MPI_STATUS_IGNORE);
}
```

Ordinea operaţiilor de tip *Send/Receive* este importantă întrucât este posibilă apariţia unor situaţii de *deadlock*. În exemplul următor fiecare proces execută mai întâi o operaţie de tip *Receive*, astfel încât fiecare va aştepta date indefinit.

```
if (rank==0) {
    MPI_Recv( recv_data2, n, MPI_INT, 1, 0, MPI_COMM_WORLD,
        MPI_STATUS_IGNORE);
    MPI_Send( send_data1, n, MPI_INT, 1, 0, MPI_COMM_WORLD);
}
if (rank==1) {
    MPI_Recv( recv_data1, n, MPI_INT, 0, 0, MPI_COMM_WORLD,
        MPI_STATUS_IGNORE);
    MPI_Send( send_data2, n, MPI_INT, 0, 0, MPI_COMM_WORLD);
}
```

În funcţie de implementare, se pot folosi buffere de sistem, iar datele stocate pot fi preluate ulterior. În exemplul următor, operaţiile *Send* sunt de tip blocking. Cu toate acestea, situaţia de deadlock este evitată dacă sunt utilizate bufferele sistemului.

```
if (rank==0) {
    MPI_Send( send_data1, n, MPI_INT, 1, 0, MPI_COMM_WORLD);
    MPI_Recv( recv_data2, n, MPI_INT, 1, 0, MPI_COMM_WORLD,
        MPI_STATUS_IGNORE);
```

```
}
if (rank==1) {
    MPI_Send( send_data2, n, MPI_INT, 0, 0, MPI_COMM_WORLD);
    MPI_Recv( recv_data1, n, MPI_INT, 0, 0, MPI_COMM_WORLD,
        MPI_STATUS_IGNORE);
}
```

Un fragment de cod care nu produce situaţii de deadlock, chiar dacă nu sunt folosite bufferele sistemului este următorul:

```
if (rank==0) {
    MPI_Send( send_data1, n, MPI_INT, 1, 0, MPI_COMM_WORLD);
    MPI_Recv( recv_data2, n, MPI_INT, 1, 0, MPI_COMM_WORLD,
        MPI_STATUS_IGNORE);
}
if (rank==1) {
    MPI_Recv( recv_data1, n, MPI_INT, 0, 0, MPI_COMM_WORLD,
        MPI_STATUS_IGNORE);
    MPI_Send( send_data2, n, MPI_INT, 0, 0, MPI_COMM_WORLD);
}
```

În general, o implementare sigură se poate obţine dacă rangurile pare execută *Send/Receive*, iar cele impare *Receive/Send*.