```c
#include <stdio.h>
#include <string.h>
#include "mpi.h"

int main(int argc, char** argv) {
    int         my_rank;       /* rank of process      */
    int         rank_x_10;      /* rank of process times 10      */
    int         received_rank_x_10;      /* rank of process times 10      */
    int         p;             /* number of processes  */
    int         source;        /* rank of sender       */
    int         dest;          /* rank of receiver     */
    int         tag = 0;       /* tag for messages     */
    char        message[100];  /* storage for message  */
    MPI_Status  status;        /* return status for    */
                               /* receive              */
    MPI_Request request;

    /* Start up MPI */
    MPI_Init(&argc, &argv);

    /* Find out process rank  */
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    /* Find out number of processes */
    MPI_Comm_size(MPI_COMM_WORLD, &p);

    if (my_rank != 0) {
        /* Create message */
        dest = 0;
        rank_x_10 = my_rank * 10 ; // supercalcolo LOL!
        MPI_Send(&rank_x_10, 1, MPI_INT, dest, tag, MPI_COMM_WORLD);

    } else { /* my_rank == 0 */
        for (source = 1; source < p; source++) {
            // Provare con e sneza tag MPI_ANY_SOURCE ;)
            //MPI_Recv(&received_rank_x_10, 1, MPI_INT, source, tag, MPI_COMM_WORLD, &status)
            //printf("Received %d rank10 from source %d\n", received_rank_x_10, source);
            MPI_Recv(&received_rank_x_10, 1, MPI_INT, MPI_ANY_SOURCE, tag, MPI_COMM_WORLD, &s
            printf("Received %d rank10 from source %d\n", received_rank_x_10, status.MPI_SOUR
        }
    }

    /* Shut down MPI */
    MPI_Finalize();
} /* main */

```