```c
#include <stdio.h>
#include "mpi.h"
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>


#define MAXTASKS     8192
/* Change the next four parameters to suit your case */
#define STARTSIZE     1000000
#define ENDSIZE       1000000
#define INCREMENT     100000
#define ROUNDTRIPS    100


int main(int argc,char *argv[]){
    int     numtasks, rank, n, i, j, rndtrps, nbytes, start, end, incr,
    src, dest, rc, tag=1, taskpairs[MAXTASKS], namelength;

    double  thistime, bw, bestbw, worstbw, totalbw, avgbw,
    bestall, avgall, worstall, timings[MAXTASKS/2][3],
    tmptimes[3], resolution, t1, t2;

    char msgbuf[ENDSIZE], host[MPI_MAX_PROCESSOR_NAME],
    hostmap[MAXTASKS][MPI_MAX_PROCESSOR_NAME];

    MPI_Status status;

    /* Some initializations and error checking */
    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
    if (numtasks % 2 != 0) {
        printf("ERROR: Must be an even number of tasks!  Quitting... \n");
        MPI_Abort(MPI_COMM_WORLD, rc);
        exit(0);
    }

    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    start = STARTSIZE; end = ENDSIZE;
    incr = INCREMENT; rndtrps = ROUNDTRIPS;
    for (i=0; i<end; i++)  msgbuf[i] = 'x';

    /* All tasks send their host name to task 0 */
    MPI_Get_processor_name(host, &namelength);
    MPI_Gather(&host, MPI_MAX_PROCESSOR_NAME, MPI_CHAR, &hostmap, MPI_MAX_PROCESSOR_NAME, MPI_CHAR, 0, MPI_COMM_WORLD);

    /* Determine who my send/receive partner is and tell task 0 */
    if (rank < numtasks/2)   dest = src = numtasks/2 + rank;
    if (rank >= numtasks/2)  dest = src = rank - numtasks/2;
    MPI_Gather(&dest,1,MPI_INT,&taskpairs,1,MPI_INT,0,MPI_COMM_WORLD);

    if (rank == 0) {
        resolution = MPI_Wtick();
        printf("\n****************** MPI/POE Bandwidth Test****\n");
        printf("Message start size= %d bytes\n",start);
        printf("Message finish size= %d bytes\n",end);
        printf("Incremented by %d bytes per iteration\n",incr);
        printf("Roundtrips per iteration= %d\n",rndtrps);
        printf("MPI_Wtick resolution = %e\n",resolution);
        printf("*****************************************************\n");
        for (i=0; i<numtasks; i++)
            printf("task %4d is on %s partner=%4d\n",i,hostmap[i],taskpairs[i]);
        printf("*****************************************************\n");
    }

    /****** first half of tasks **************/
    /* These tasks send/receive messages with their partner task, */
    /* and then do a few bandwidth calculations based upon */
    /* message size and timings.           */
    if (rank < numtasks/2) {
        for (n=start; n<=end; n=n+incr) {
            bestbw = 0.0; worstbw = .99E+99; totalbw = 0.0;
            nbytes =  sizeof(char) * n;
            for (i=1; i<=rndtrps; i++){
                t1 = MPI_Wtime();
                MPI_Send(&msgbuf,n,MPI_CHAR,dest,tag,MPI_COMM_WORLD);
                MPI_Recv(&msgbuf,n,MPI_CHAR,src, tag,MPI_COMM_WORLD,&status);
                t2 = MPI_Wtime();
                thistime = t2 - t1;
                bw = ((double)nbytes * 2) / thistime;
                totalbw = totalbw + bw;
                if (bw > bestbw ) bestbw = bw;
                if (bw < worstbw ) worstbw = bw;
            }
            /* Convert to megabytes per second */
            bestbw = bestbw/1000000.0; worstbw = worstbw/1000000.0;
            avgbw = (totalbw/1000000.0)/(double)rndtrps;

            if(rank == 0) {
                /* Task 0 collects timings from all relevant tasks */
                /* Keep track of my own timings first */
                timings[0][0] = bestbw;  timings[0][1] = avgbw;  timings[0][2] = worstbw;
                bestall = 0.0;    avgall = 0.0;      worstall = 0.0;
                /* Initialize overall averages */
                /* Now receive timings from other tasks and print results. */
                for (j=1; j<numtasks/2; j++)
                    MPI_Recv
                    (&timings[j], 3, MPI_DOUBLE, j, tag, MPI_COMM_WORLD, &status);
                printf("***Message size: %8d *** best  /  avg  / worst (MB/sec)\n",n);
                for (j=0; j<numtasks/2; j++) {
                    printf("   task pair: %4d - %4d:    %4.2f / %4.2f / %4.2f \n", j, taskpairs[j],timings[j][0], timings[j][1], timings[j][2]);
                    bestall += timings[j][0];  avgall += timings[j][1]; worstall += timings[j][2];
                }
                printf("   OVERALL AVERAGES:          %4.2f / %4.2f / %4.2f \n\n",
                       bestall/(numtasks/2), avgall/(numtasks/2), worstall/(numtasks/2));
            }
            else {
                /* Other tasks send their timings to task 0 */
                tmptimes[0] = bestbw;    tmptimes[1] = avgbw;    tmptimes[2] = worstbw;
                MPI_Send
                (tmptimes, 3, MPI_DOUBLE, 0, tag, MPI_COMM_WORLD);
            }
        }
    }

    /*************** second half of tasks ***************************/
    /* These tasks do nothing more than send and receive with their partner task */
    if (rank >= numtasks/2) {
        for (n=start; n<=end; n=n+incr) {
            for (i=1; i<=rndtrps; i++){
                MPI_Recv(&msgbuf, n, MPI_CHAR, src, tag, MPI_COMM_WORLD, &status);
                MPI_Send(&msgbuf, n, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
            }
        }
    }
    MPI_Finalize();


    return 0;
}
```