

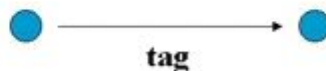
Comunicación Bloqueante y no Bloqueante

Comunicación Bloqueante	Comunicación no Bloqueante
En el envío común de mensajes la comunicación bloqueante no realiza una acción hasta que haya terminado otra, es decir, si se manda un mensaje, no se realiza otra acción en el programa hasta recibir una respuesta	Bajo el mismo caso, la comunicación no bloqueante realiza las acciones posteriores al envío y respuesta de inmediato; es por esto que pueden ocurrir errores, pero a su vez se incrementa enormemente la velocidad. Para evitar estos errores debe revisarse el estado del buffer si es que se terminó de realizar la acción que requiere del envío.

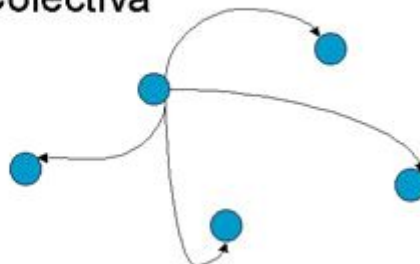
Tanto el método de comunicación Bloqueante como no Bloqueante se pueden aplicar en la comunicación de Punto a Punto como Colectiva; dependiendo del tipo de comunicación y su cantidad de nodos esta puede llegar a ser mucho más sensible a fallas en el caso de la no Bloqueante.

Tipos de comunicación

■ Punto a punto



■ Colectiva



Comunicaciones punto a punto

Se da entre pares de procesos, la transmisión de mensajes se da por medio de un ranking y un tag.

Por otra parte también se pueden haber estos modos de comunicación dentro de la comunicación punto a punto:

1. Envío Estandar
Se retorna una respuesta después de haberse enviado.
2. Envío Síncrono
La llamada termina cuando el mensaje fue recibido.
3. Buffered send
Termina inmediatamente, pero copia el mensaje en un buffer en caso de necesitarlo posteriormente.
4. Ready Send
Termina inmediatamente, se utiliza un recibidor antes del envío, este es exitoso solo al invocarse.

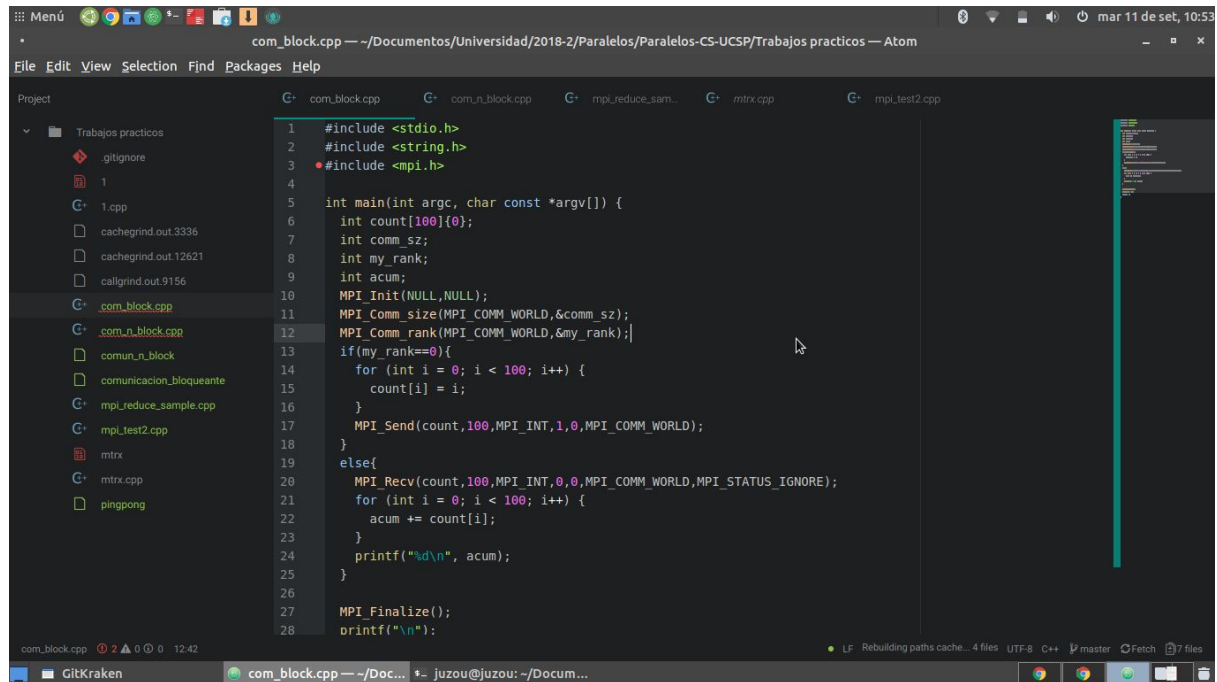
Comunicaciones colectivas

Comunicación coordinada entre un grupo de procesos donde no se usan tags; todos los procesos están entrelazados a un comunicador, dentro de esta comunicación se puede crear otra sub-comunicación pero se debe establecer un nuevo comunicador; todas las rutinas se bloquean hasta completarse localmente; se puede establecer una comunicación de datos o de computación global.

Ejemplo de comunicación Bloqueante

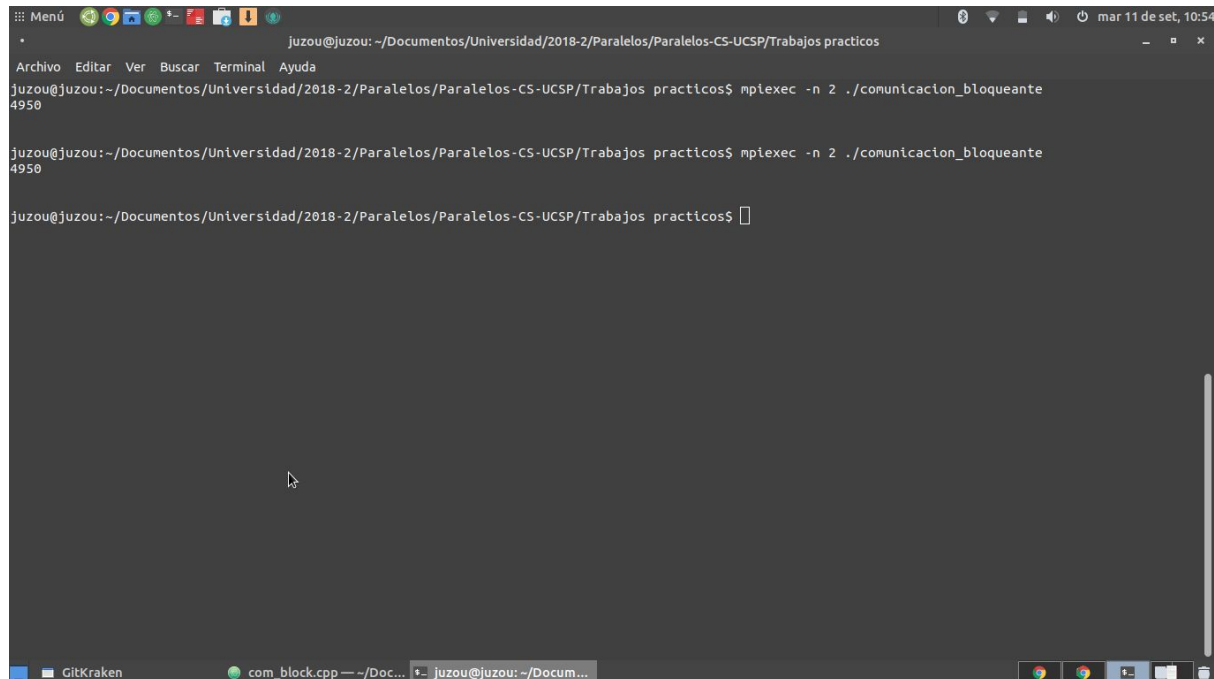
En el caso de la comunicación Bloqueante se espera a recibir los datos enviados por el proceso padre; luego de recibirlos se empieza a operar sobre ellos.

Codigo



```
1 #include <stdio.h>
2 #include <string.h>
3 #include <mpi.h>
4
5 int main(int argc, char const *argv[]) {
6     int count[100]{0};
7     int comm_sz;
8     int my_rank;
9     int acum;
10    MPI_Init(NULL, NULL);
11    MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
12    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
13    if(my_rank == 0){
14        for (int i = 0; i < 100; i++) {
15            count[i] = i;
16        }
17        MPI_Send(count, 100, MPI_INT, 1, 0, MPI_COMM_WORLD);
18    }
19    else{
20        MPI_Recv(count, 100, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
21        for (int i = 0; i < 100; i++) {
22            acum += count[i];
23        }
24        printf("%d\n", acum);
25    }
26
27    MPI_Finalize();
28    printf("\n");
}
```

Salida

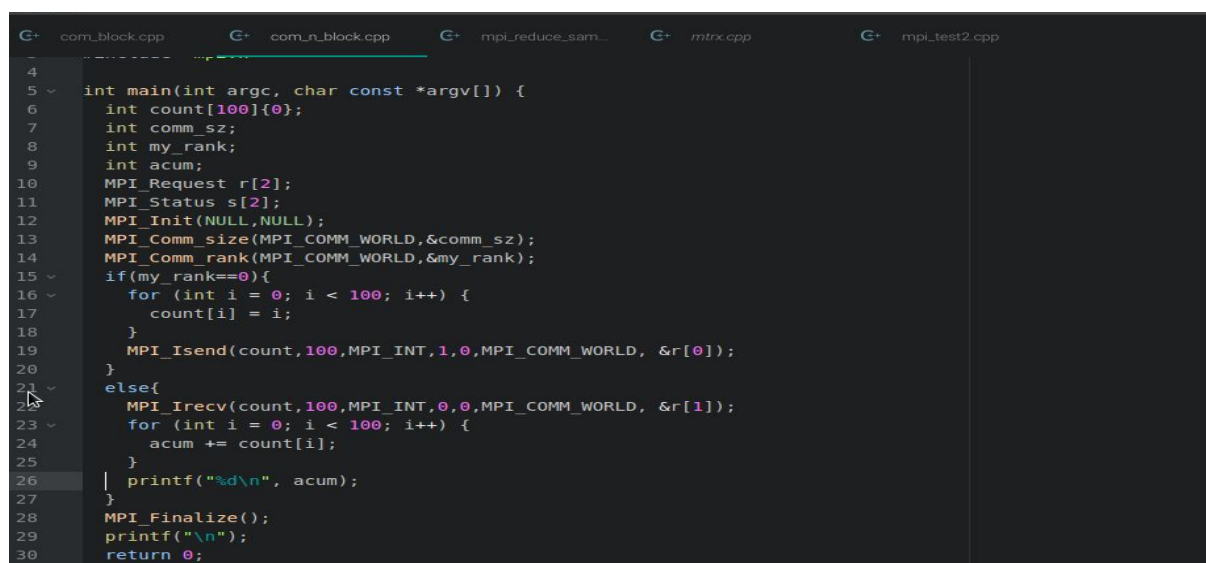


```
juzou@juzou: ~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos
Archivo Editar Ver Buscar Terminal Ayuda
juzou@juzou: ~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos$ mpiexec -n 2 ./comunicacion_bloqueante
4950
juzou@juzou: ~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos$ mpiexec -n 2 ./comunicacion_bloqueante
4950
juzou@juzou: ~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos$
```

Ejemplo de comunicación no Bloqueante

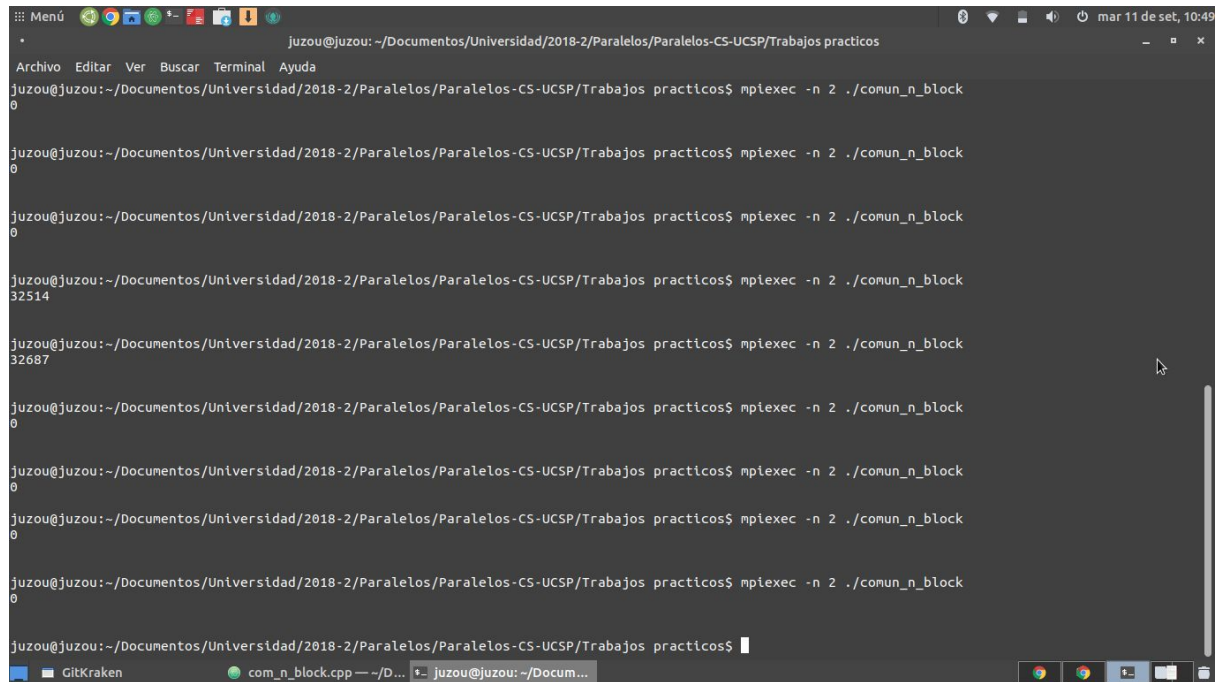
Para la comunicación no bloqueante existen dos funciones de mpi, `MPI_Isend` y `MPI_Irecv`, ambas teniendo como ultimo valor un `MPI_Request` como valor enviado, dato que puede ser utilizado para esperar antes de recibir los datos; como se ve en la salida no siempre se recibirán la totalidad de los datos o una parte, ya que en el MPI no bloqueante no se espera a nadie.

Código



```
com_block.cpp com_n_block.cpp mpi_reduce_sam... mtrx.cpp mpi_test2.cpp
4
5 int main(int argc, char const *argv[]) {
6     int count[100]{0};
7     int comm_sz;
8     int my_rank;
9     int acum;
10    MPI_Request r[2];
11    MPI_Status s[2];
12    MPI_Init(NULL, NULL);
13    MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
14    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
15    if(my_rank==0){
16        for (int i = 0; i < 100; i++) {
17            count[i] = i;
18        }
19        MPI_Isend(count, 100, MPI_INT, 1, 0, MPI_COMM_WORLD, &r[0]);
20    }
21    else{
22        MPI_Irecv(count, 100, MPI_INT, 0, 0, MPI_COMM_WORLD, &r[1]);
23        for (int i = 0; i < 100; i++) {
24            acum += count[i];
25        }
26        printf("%d\n", acum);
27    }
28    MPI_Finalize();
29    printf("\n");
30    return 0;
31}
```

Salida



```
juzou@juzou: ~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos
Archivo Editar Ver Buscar Terminal Ayuda
juzou@juzou:~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos$ mpirun -n 2 ./comun_n_block
0

juzou@juzou:~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos$ mpirun -n 2 ./comun_n_block
0

juzou@juzou:~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos$ mpirun -n 2 ./comun_n_block
0

juzou@juzou:~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos$ mpirun -n 2 ./comun_n_block
32514

juzou@juzou:~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos$ mpirun -n 2 ./comun_n_block
32687

juzou@juzou:~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos$ mpirun -n 2 ./comun_n_block
0

juzou@juzou:~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos$ mpirun -n 2 ./comun_n_block
0

juzou@juzou:~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos$ mpirun -n 2 ./comun_n_block
0

juzou@juzou:~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos$ mpirun -n 2 ./comun_n_block
0

juzou@juzou:~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos$
```

Referencias

<https://users.dcc.uchile.cl/~ynakanis/pagina/presenta.html#punto>