3.3.
Communications MPI point à point

3 – Communications point à point
3.1 – Notions générales

Notions générales

Une communication dite point à point a lieu entre deux processus, l'un appelé processus émetteur et l'autre processus récepteur (ou destinataire).

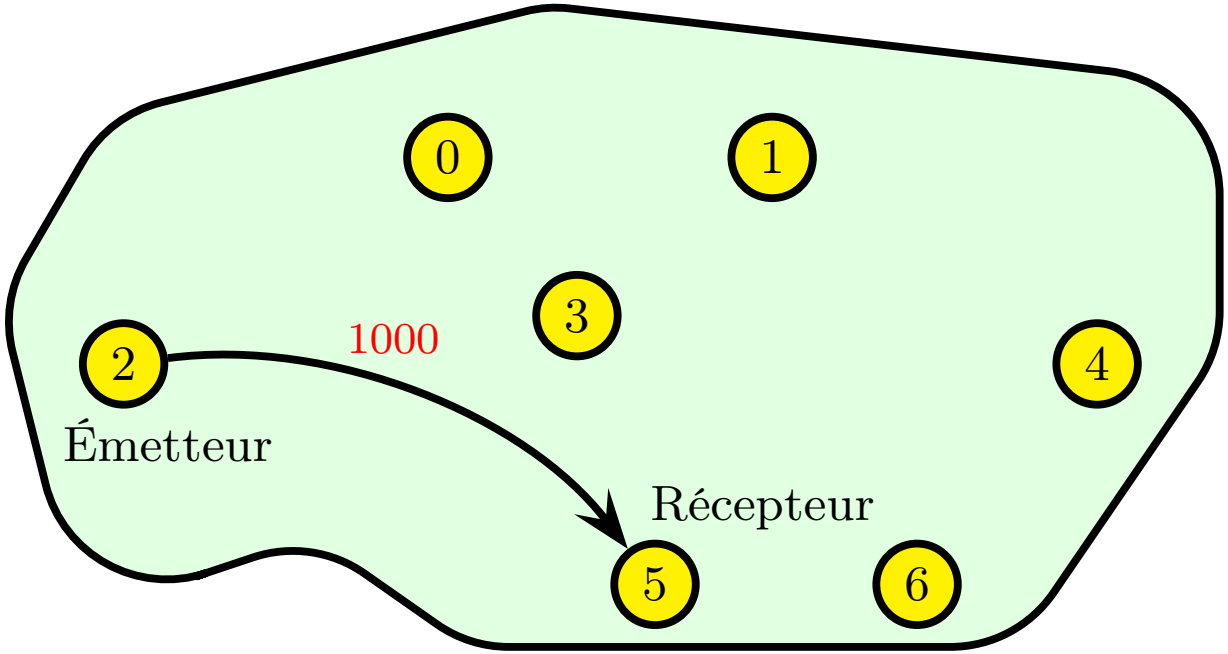

FIGURE 10 – Communication point à point

Notions générales

- L'émetteur et le récepteur sont identifiés par leur rang dans le communicateur.
- Ce que l'on appelle l'enveloppe d'un message est constituée :
  - du rang du processus émetteur ;
  - du rang du processus récepteur ;
  - de l'étiquette (*tag*) du message ;
  - du communicateur qui définit le groupe de processus et le contexte de communication.
- Les données échangées sont typées (entiers, réels, etc ou types dérivés personnels).
- Il existe dans chaque cas plusieurs modes de transfert, faisant appel à des protocoles différents.

3 – Communications point à point
3.2 – Opérations d'envoi et de réception bloquantes

Opération d'envoi MPI_SEND

```
<type et attribut>:: message
integer :: longueur, type
integer :: rang_dest, etiquette, comm, code
call MPI_SEND(message,longueur,type,rang_dest,etiquette,comm,code)
```

Envoi, à partir de l'adresse message, d'un message de taille longueur, de type type, étiqueté etiquette, au processus rang_dest dans le communicateur comm.

Remarque :
Cette opération est bloquante : l'exécution reste bloquée jusqu'à ce que le contenu de message puisse être réécrit sans risque d'écraser la valeur qui devait être envoyée.

Opération de réception MPI_RECV

```
<type et attribut>:: message
integer :: longueur, type
integer :: rang_source, etiquette, comm, code
integer, dimension(MPI_STATUS_SIZE) :: statut
call MPI_RECV(message,longueur,type,rang_source,etiquette,comm,statut,code)
```

Réception, à partir de l'adresse message, d'un message de taille longueur, de type type, étiqueté etiquette, du processus rang_source.

Remarques :

- statut reçoit des informations sur la communication : rang_source, etiquette, code, ...
- L'appel MPI_RECV ne pourra fonctionner avec une opération MPI_SEND que si ces deux appels ont la même enveloppe (rang_source, rang_dest, etiquette, comm).
- Cette opération est bloquante : l'exécution reste bloquée jusqu'à ce que le contenu de message corresponde au message reçu.

```
1  program point_a_point
2    use mpi
3    implicit none
4
5    integer, dimension(MPI_STATUS_SIZE) :: statut
6    integer, parameter                  :: etiquette=100
7    integer                             :: rang,valeur,code
8
9    call MPI_INIT(code)
10
11   call MPI_COMM_RANK(MPI_COMM_WORLD,rang,code)
12
13   if (rang == 2) then
14      valeur=1000
15      call MPI_SEND(valeur,1,MPI_INTEGER,5,etiquette,MPI_COMM_WORLD,code)
16   elseif (rang == 5) then
17      call MPI_RECV(valeur,1,MPI_INTEGER,2,etiquette,MPI_COMM_WORLD,statut,code)
18      print *,'Moi, processus 5, j''ai reçu ',valeur,' du processus 2.'
19   end if
20
21   call MPI_FINALIZE(code)
22
23 end program point_a_point
```

```
> mpiexec -n 7 point_a_point

Moi, processus 5, j'ai reçu 1000 du processus 2
```

3 – Communications point à point
3.3 – Types de données de base

Types de données de base Fortran

Table 1 – Principaux types de données de base (Fortran)

Type MPI	Type Fortran
MPI_INTEGER	`INTEGER`
MPI_REAL	`REAL`
MPI_DOUBLE_PRECISION	`DOUBLE PRECISION`
MPI_COMPLEX	`COMPLEX`
MPI_LOGICAL	`LOGICAL`
MPI_CHARACTER	`CHARACTER`

Types de données de base C

Table 2 – Principaux types de données de base (C)

Type MPI	Type C
MPI_CHAR	`signed char`
MPI_SHORT	`signed short`
MPI_INT	`signed int`
MPI_LONG	`signed long int`
MPI_UNSIGNED_CHAR	`unsigned char`
MPI_UNSIGNED_SHORT	`unsigned short`
MPI_UNSIGNED	`unsigned int`
MPI_UNSIGNED_LONG	`unsigned long int`
MPI_FLOAT	`float`
MPI_DOUBLE	`double`
MPI_LONG_DOUBLE	`long double`

3.4.

Phénomène de « deadlock » avec MPI

Deadlock ou inter-blocage

2 tâches MPI veulent échanger des messages, mais toutes les 2 veulent envoyer leur message respectif et ne sont pas prêtes à recevoir.

L'exécution d'un programme faisant appel à la fonction MPI_Send suivie de MPI_Recv peut rester bloquée sur MPI_Send quant deux processus s'échangent des messages de tailles importantes.

Ce problème est dû à l'utilisation du mode d'envoi MPI dit standard MPI_Send. Ce mode autorise l'implémentation MPI à choisir la façon d'envoyer les messages.

En général, les petits messages sont recopiés dans un espace mémoire temporaire avant d'être expédiés (bufferisés).

Les gros messages sont envoyés en mode synchrone. Ce mode synchrone implique que pour que le message parte (càd pour que l'appel MPI_Send puisse se terminer), il faut que la réception de ce message ait été postée (càd que l'appel à MPI_Recv ait été effectué).

Deadlock ou inter-blocage

1er scénario possible :

Process 0	Process 1
Send(1)	Send(0)
Recv(1)	Recv(0)

Ca peut marcher pour les envois de petite taille qui seront « bufferisés » mais pas pour les envois de grandes taille qui fonctionnent en mode synchrone.

Ce schéma de communication n'est pas du tout conseillé et est d'ailleurs considéré comme erroné par la norme MPI.

La valeur par défaut de la taille à partir de laquelle les envois ne sont pas « bufferisés » (donc synchrones) varie d'une architecture de machine à une autre. Cette valeur peut être changée selon les architectures.

Si vous voulez vous assurer que votre application ne risque pas de souffrir de ce problème, il est conseillé de la tester en mettant cette valeur à 0 afin de forcer le mode synchrone pour tous les envois standards. Si tout se passe bien (pas de blocage), votre application ne devrait pas avoir ce souci.

Deadlock ou inter-blocage

2ème scénario possible :

Process 0	Process 1
Recv(1)	Recv(0)
Send(1)	Send(0)

Ca bloque définitivement quelque soit la taille des messages envoyés.

Les deux processus sont bloqués au Recv et les Send ne s'exécutent pas.

Deadlock ou inter-blocage

Le 3ème scénario qui évite le phénomène de deadlock :

Process 0	Process 1
Send(1)	Recv(0)
Recv(1)	Send(0)

Attention aux étiquettes : si un processus de rang R1 envoie un message M au processus R2 avec l'étiquette E1, le processus R2 reçoit le message M avec la même étiquette E1. Même avec ce scénario, si les étiquettes ne sont pas bien mises, ça peut bloquer quand même !