LES COMMUNICATIONS

APP– 3 Ing Info Systèmes Embarqués

Chapitre 2

Plan

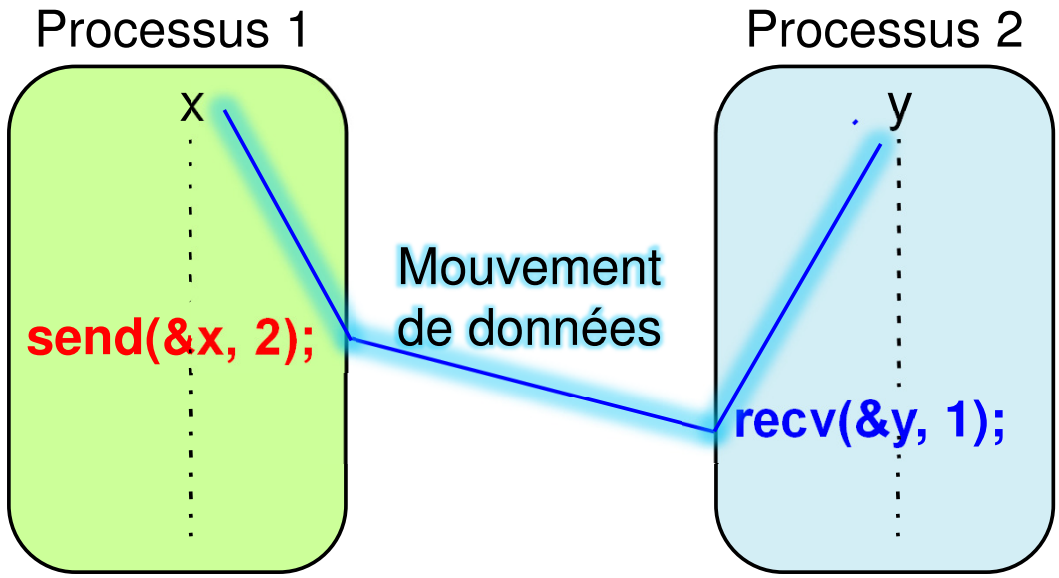
1. Communication point à point
2. Macro - communication
3. Evaluation des coûts
4. Bibliothèques de communication
5. Outils d’analyse de performances

Communication point à point

1. Les envois et les réceptions

Un passage de messages entre processus utilise les appels systèmes

send() et **recv()**



Communication point à point

2. Le modèle synchrone

Synchronous send routine (routine d'envoi synchrone)

- **Routines ne retournent qu'après que le transfert de message soit fini.**

Synchronous receive routine (routine de réception synchrone)

- **attend jusqu'à ce que la totalité du message soit acceptée par le processus récepteur**

- **Les routines synchrones performent d'une façon intrinsèque deux actions:**
  - **elles transfèrent les données**
  - **et elles synchronisent les processus.**

Communication point à point

2. Le modèle synchrone

Synchronous send routine (routine d'envoi synchrone)

- **Routines ne retournent qu'après que le transfert de message soit fini.**

Synchronous receive routine (routine de réception synchrone)

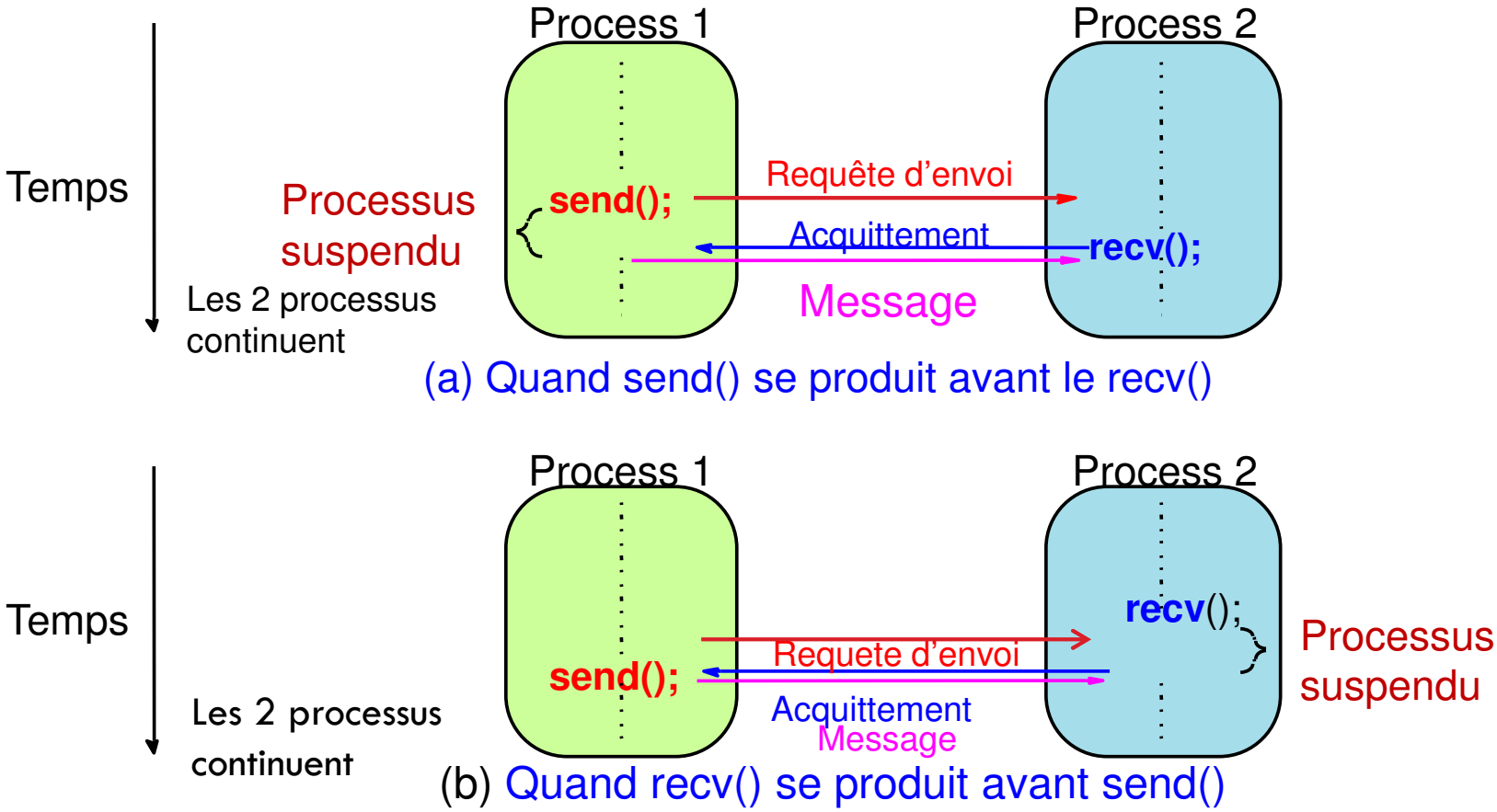
- **attend jusqu'à ce que la totalité du message soit acceptée par le processus récepteur**

+ Message reçu

- Temps d'attente

Communication point à point

2. Le modèle synchrone – 3 way protocol



Communication point à point

3. Le modèle asynchrone

- **Routines qui n'attendent pas la fin des actions avant de retourner. D'ordinaire, elles exigent un stockage local des messages.**

- **En général, elles ne synchronisent pas les processus mais permettent aux processus d'avancer le plus vite possible.**

- **Doivent être utilisées avec précaution.**

+ Pas de temps d'attente

- Message reçu? (Sol: accusé de réception)

Communication point à point

4. Le modèle bloquant (Blocking)

Communication asynchrone

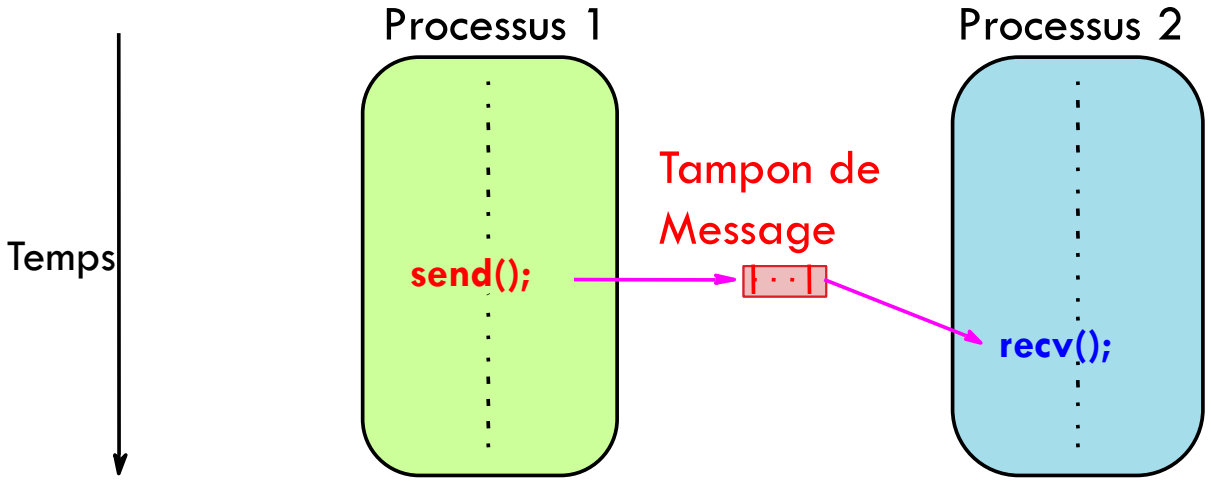
L'envoi ou la réception se termine après que leurs actions locales sont finies, même si le transfert du message n'est pas encore fini.

Communication point à point

5. Le modèle non bloquant (Non-blocking)

Communication asynchrone

- **processus retournent immédiatement.**
- **Assume que l'espace de stockage utilisé pour le transfert n'est pas modifié par les commandes qui suivent, jusqu'à ce que le transfert soit fini.**



Les macro-communications

- **Macro-communications = Communications collectives**
- **Concernent un sous-ensemble de processus et nécessitent une synchronisation**

Les macro-communications

1. Les primitives

- Des routines qui permettent l'envoi ou la réception d'un ou plusieurs messages entre un groupe de processus

- Ces routines offrent une efficacité plus élevée que les routines d'action point à point

Les macro-communications

1. Les primitives

❖ La diffusion (broadcast ou one-to-all)

Envoi du même message à tous les processus concernés

❖ La distribution (Scratting ou personalised one-to-all)

Envoi des messages différents aux autres processus

❖ Le rassemblement (Gathering)

Opération inverse de la distribution : un processus reçoit des messages différents des autres processus

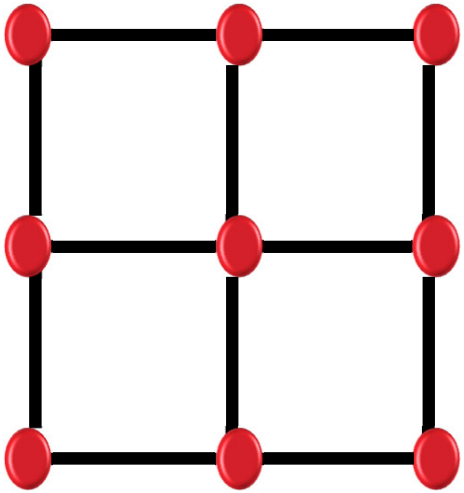
❖ La réduction (Reduction)

Synthétiser sur un processus les données obtenues des autres processus

Les macro-communications

2. Exemples

❖ **La diffusion dans une matrice (grille à deux dimensions)**

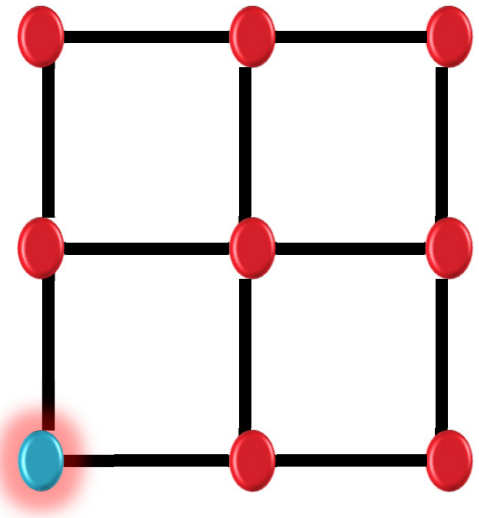


Un processus envoie le même message aux autres

Les macro-communications

2. Exemples

❖ **La diffusion dans une matrice (grille à deux dimensions)**

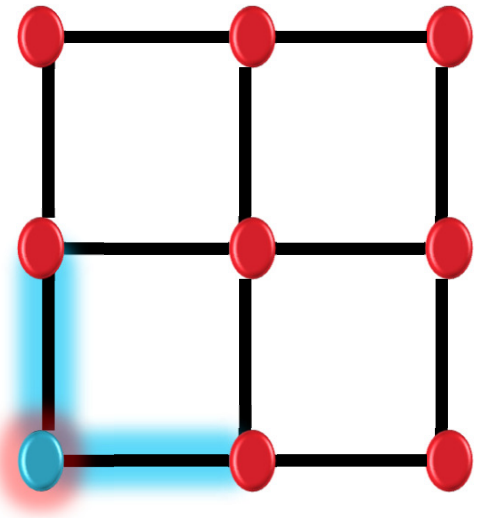


Un processus envoie le même message aux autres

Les macro-communications

2. Exemples

❖ **La diffusion dans une matrice (grille à deux dimensions)**

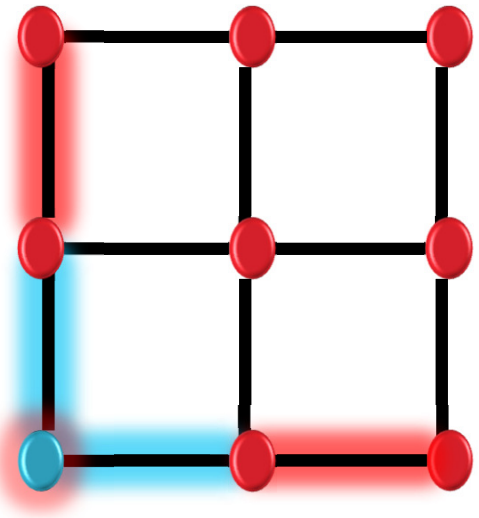


Les étapes .

Les macro-communications

2. Exemples

❖ **La diffusion dans une matrice (grille à deux dimensions)**

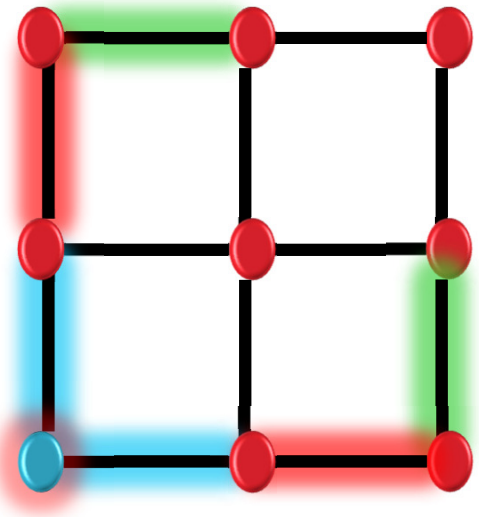


Les étapes ..

Les macro-communications

2. Exemples

❖ **La diffusion dans une matrice (grille à deux dimensions)**

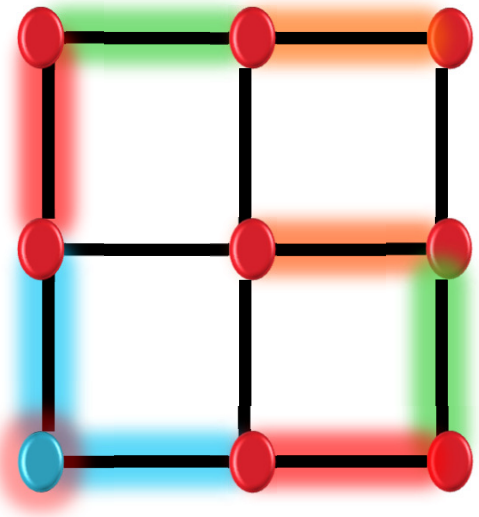


Les étapes …

Les macro-communications

2. Exemples

❖ **La diffusion dans une matrice (grille à deux dimensions)**

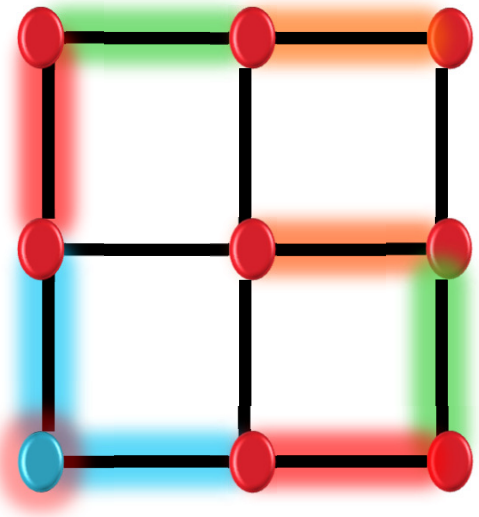


Les étapes ….

Les macro-communications

2. Exemples

❖ **La diffusion dans une matrice (grille à deux dimensions)**

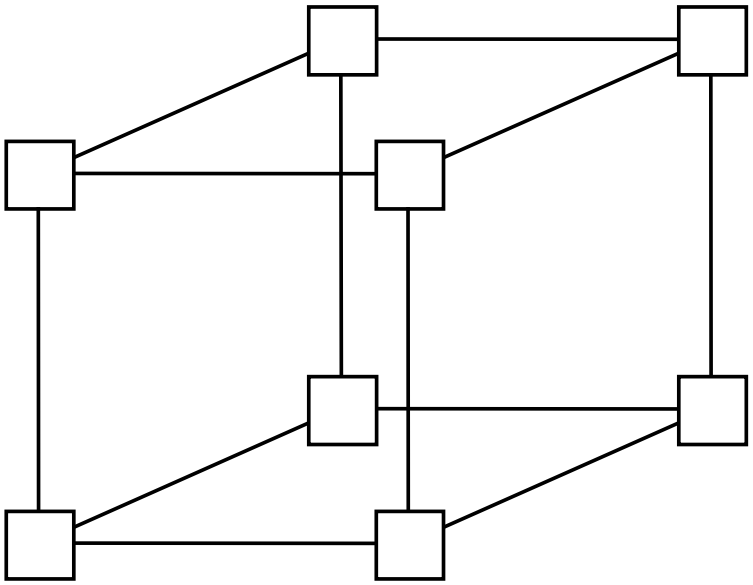


➢ **La diffusion se base sur des primitives élémentaires exécutées en parallèle et synchronisées**

Les macro-communications

2. Exemples

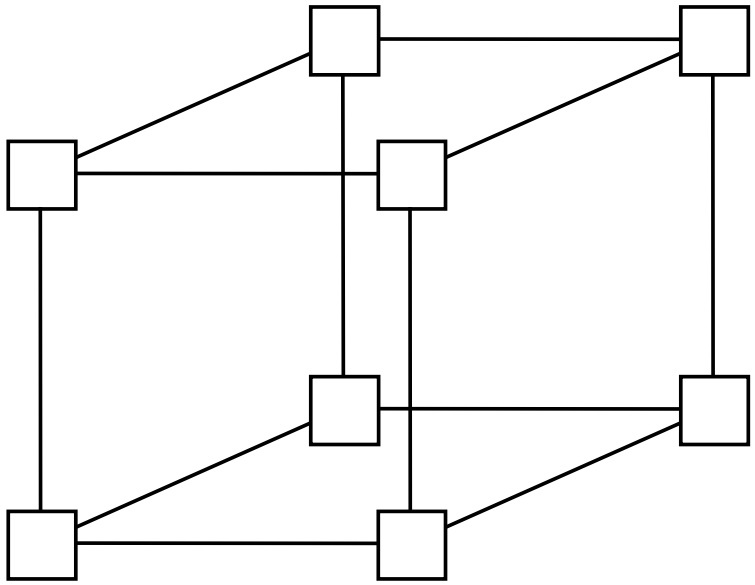
❖ **Le commérage dans un hypercube : all-to-all**



Les macro-communications

2. Exemples

❖ **Le commérage dans un hypercube : all-to-all**

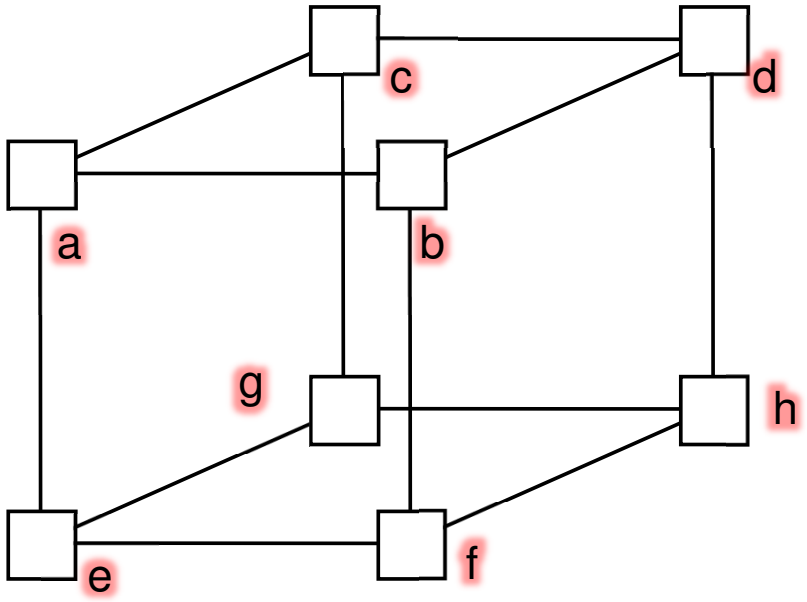


Décomposition selon une dimension et communication avec les voisins selon cette dimension

Les macro-communications

2. Exemples

❖ **Le commérage dans un hypercube : all-to-all**

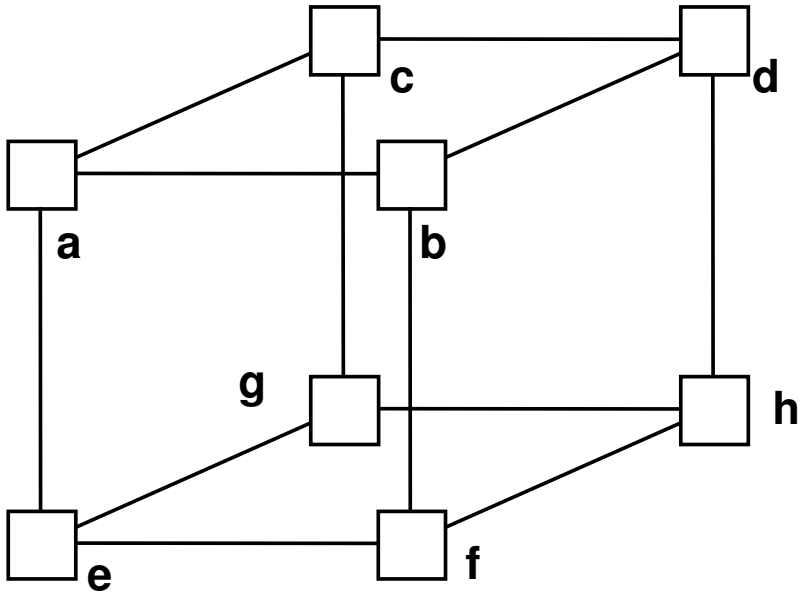


Etat initial : chaque nœud dispose d'un message

Les macro-communications

2. Exemples

❖ **Le commérage dans un hypercube : all-to-all**

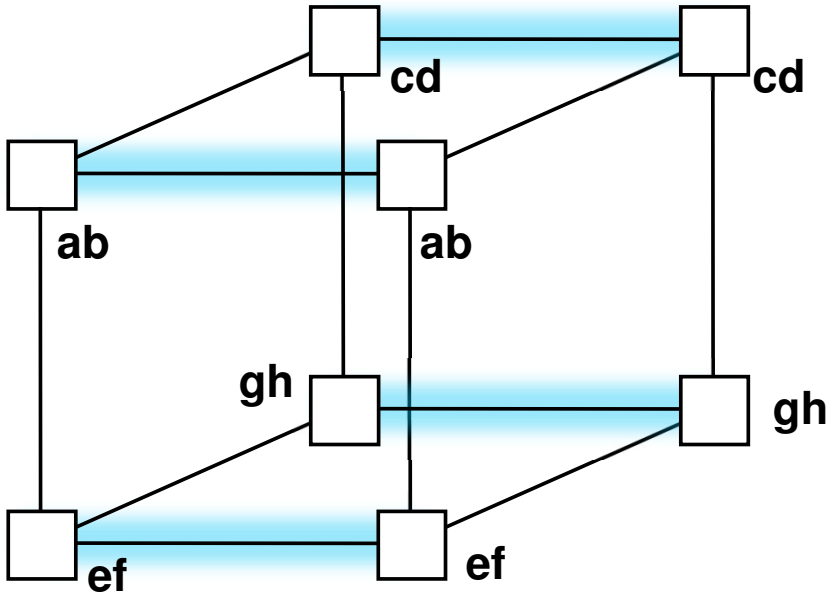


Étape 1 : transmission des messages selon le sens ⟷

Les macro-communications

2. Exemples

❖ **Le commérage dans un hypercube : all-to-all**

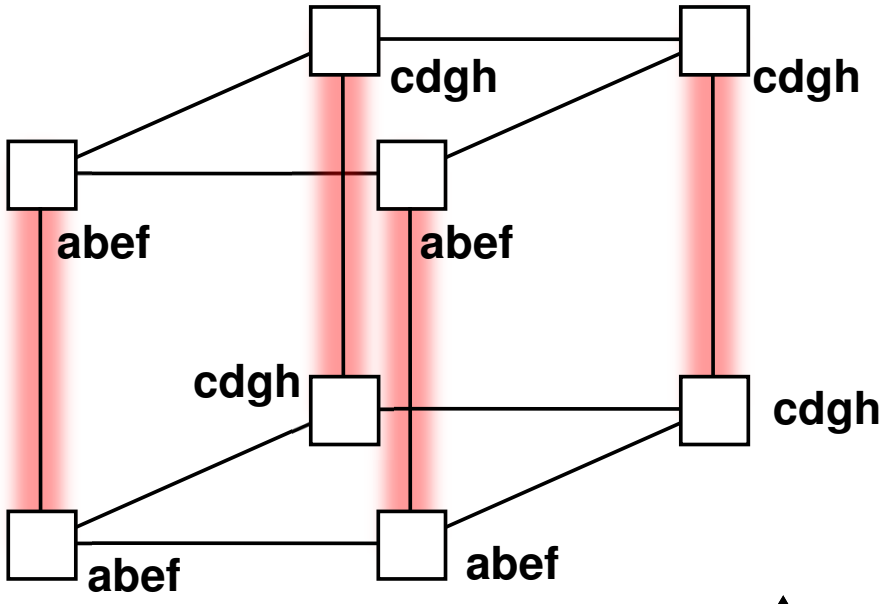


Étape 1 : transmission des messages selon le sens ⟷

Les macro-communications

2. Exemples

❖ **Le commérage dans un hypercube : all-to-all**

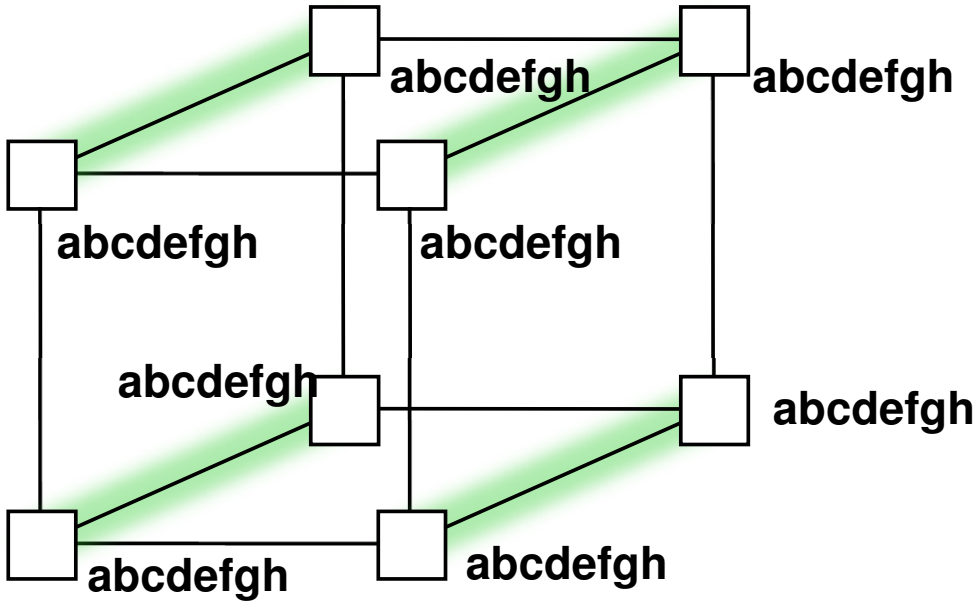


Étape 2 : transmission des messages selon le sens ↕

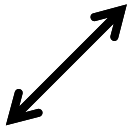
Les macro-communications

2. Exemples

❖ **Le commérage dans un hypercube : all-to-all**



Étape 3 : transmission des messages selon le sens



Evaluation des coûts

1. **Temps de communication** Le modèle le plus général pour le coût d'une communication de L données élémentaires entre deux processeurs voisins directs peut être modélisé par un temps linéaire:

$$T_{comm} = \beta + L * \tau$$

- β est le temps d'initialisation (start-up time) nécessaire pour transmettre un message au réseau de communication. Essentiellement, c'est le temps d'envoi d'un message sans donnée. il est supposé constant.

- L longueur du message en octet

- τ est le temps de transmission d'un octet.
  $1/\tau$ est la bande passante (octets/s)

Evaluation des coûts

1. Temps de communication

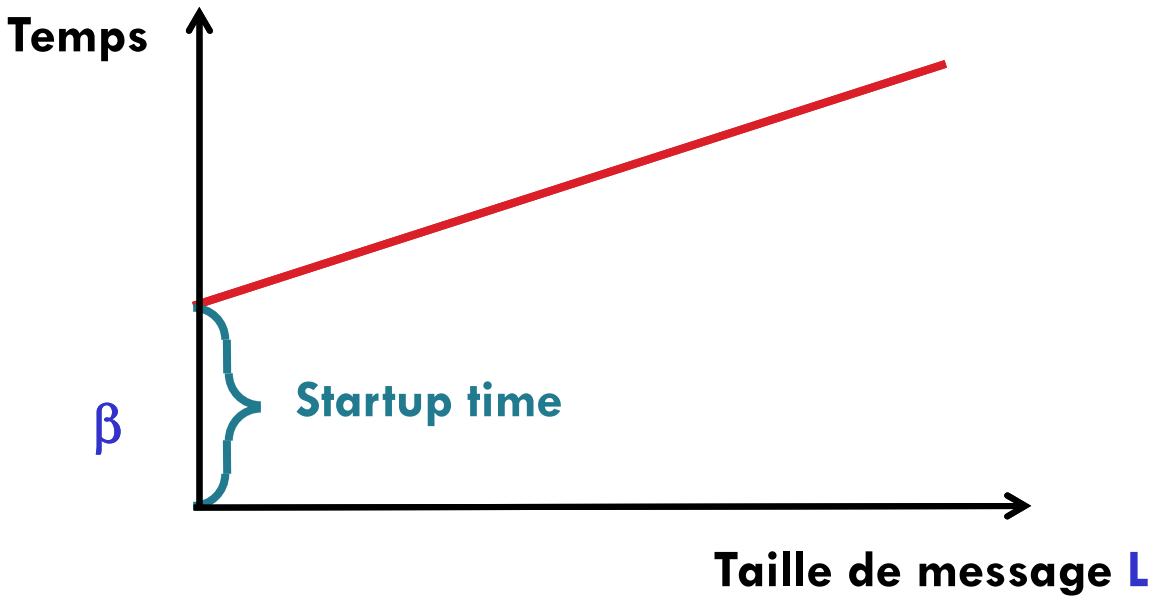
$$T_{comm} = \beta + L * \tau$$

- Sur certaines machines, comme la plupart des machines parallèles synchrones, les phases élémentaires de communication concernent les voisins directs et un message de longueur donnée est transféré en temps constant.
- Le modèle général de communication autorise des envois (et/ou réception parallèles à partir d'un même processeur. On dit alors que le *modèle est link-bound*. Cependant, on peut trouver des machines qui n'autorisent pas l'utilisation simultanée de tous les liens d'un processeurs. Ce cas correspond au modèle *processor-bound*.
- De plus, les communications sont en général bidirectionnelles (on dit que les liaisons sont *full-duplex*. Elle peuvent être unidirectionnelles (*half-duplex*).
- Sauf indication contraire, nous considérerons des liens interprocesseurs parallèles et bidirectionnels sous l'hypothèse générale du temps linéaire, ce qui est le plus courant dans les réseaux de processeurs existants.

Evaluation des coûts

1. Temps de communication

$$T_{comm} = \beta + L * \tau$$



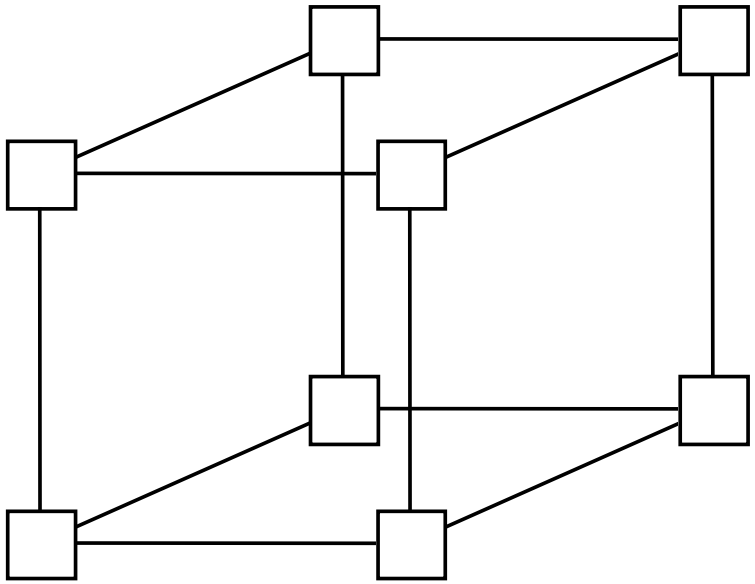
Evaluation des coûts

1. Temps de communication

$$T_{comm} = \beta + L * \tau$$

Commérage dans un hypercube

- T_{comm} = ?



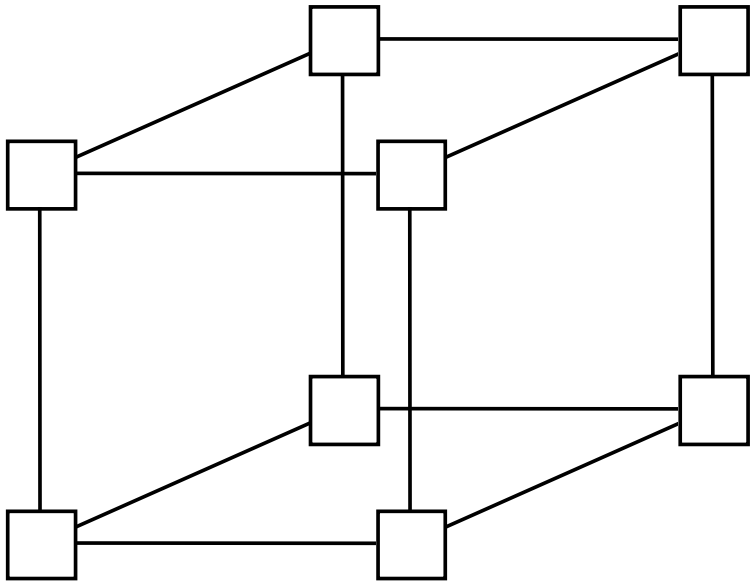
Evaluation des coûts

1. Temps de communication

$$T_{comm} = \beta + L * \tau$$

Commérage dans un hypercube

- $T_{comm} = 3* \beta + 14*L * \tau$



Evaluation des coûts

2. Facteur de Benchmark

Sur une machine parallèle

$T_p = T_{comm} + T_{calcul}$

Evaluation des coûts

2. Facteur de Benchmark

Sur une machine parallèle

$T_p = T_{comm} + T_{calcul}$

$S(p) = T_1/T_p$

$= T_1/ (T_{comm} + T_{calcul})$

❖ T_{calcul} /T_{comm} **est dit facteur de benchmark qui donne une indication sur la scalabilité : impact si on augmente le nombre de processus et la taille du problème.**

Bibliothèques de communication

1. PVM (Parallel Virtual Machine)

- Développé par Oak Ridge National Laboratories.
- Disponible gratuitement.
- Le programmeur décompose le problème en programmes séparés (souvent, un maître et un groupe de programmes esclaves identiques).
- Les programmes sont compilés pour les exécuter sur des types spécifiques d'ordinateurs.
- L'ensemble d'ordinateurs à utiliser pour un problème, doit être défini avant d'exécuter les programmes (dans un hostfile).

Bibliothèques de communication

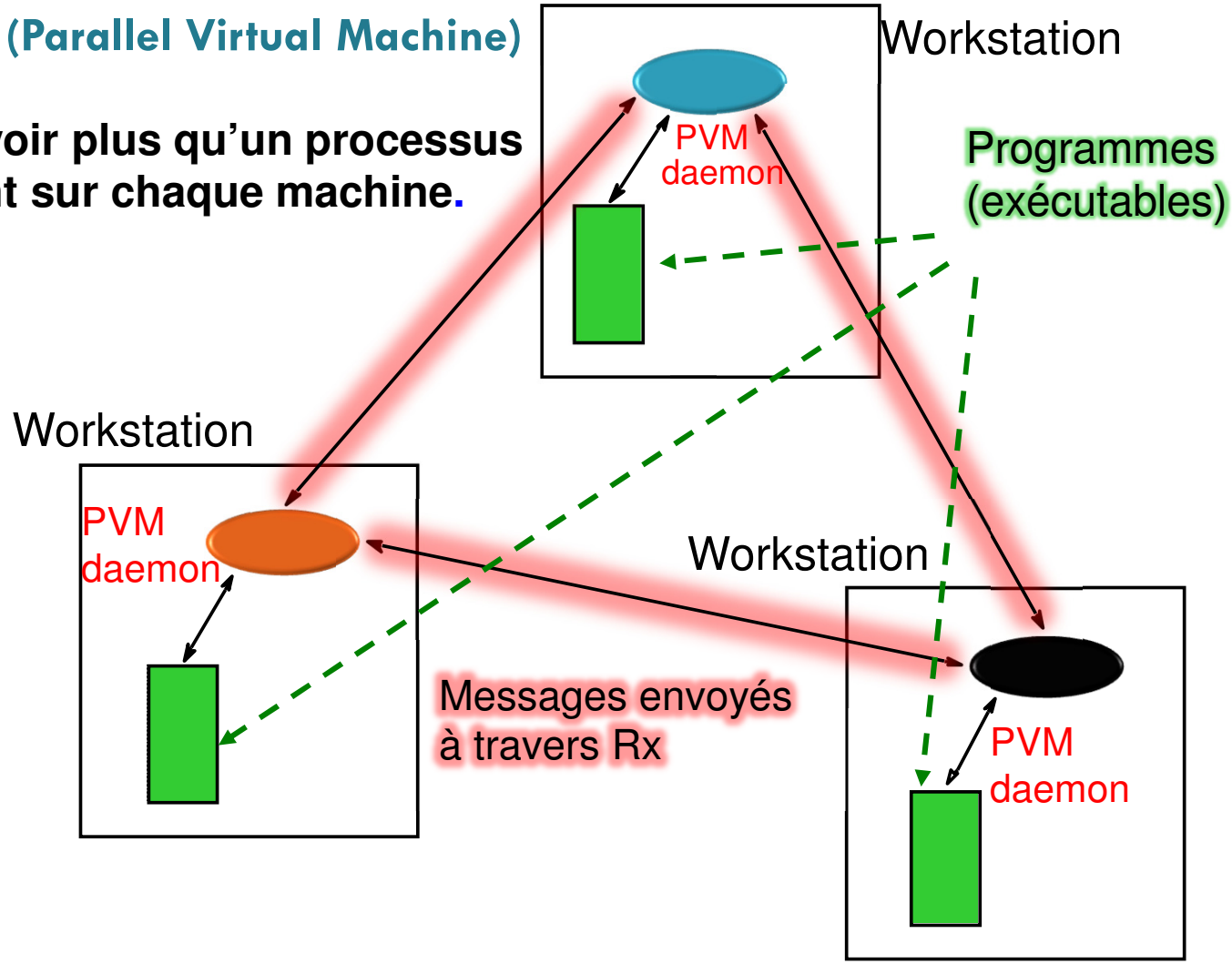
1. PVM (Parallel Virtual Machine)

- Utilisation de macro-communication.
- Envoi et réception bloquant et non-bloquant.
- Programme de type SPMD (Single Program Multiple Data) : Maître/Esclaves
- Création dynamique des processus.
- Utilise un environnement dynamique.
- Un démon est lancé avant toute exécution.

Bibliothèques de communication

1. PVM (Parallel Virtual Machine)

On peut avoir plus qu'un processus s'exécutant sur chaque machine.



Bibliothèques de communication

2. MPI (Message Passing Interface)

- **Une bibliothèque standard de passage de messages qui a été développée par des académiciens et des partenaires industriels pour promouvoir un usage large et un haut degré de portabilité.**
- **Un nombre d'implémentations gratuites existent.**

Bibliothèques de communication

2. MPI (Message Passing Interface)

Création et exécution

- Tous les processus doivent être définis avant l'exécution et commencent ensemble leur exécution.
- Modèle de calcul SPMD.
- MPMD est possible avec la création statique : les programmes qui doivent commencer ensemble sont spécifiés.

Bibliothèques de communication

2. MPI (Message Passing Interface)

Communicateur

- Définit la portée d'une opération de communication.
- Les processus possèdent des rangs associés avec le communicateur.
- Initialement, tous les processus sont admis dans un "univers" appelé MPI_COMM_WORLD, et à chaque processus est affecté un rang unique, qui est un numéro de 0 à *p* - 1, avec *p* est le nombre de processus.
- D'autres communicateurs peuvent être établis pour définir d'autres groupes de processus.

Bibliothèques de communication

2. MPI (Message Passing Interface)

Communicateur

- Définit un domaine de communication : un ensemble de processus qui sont permis de communiquer entre eux-mêmes
- Utilisés dans toutes les communications MPI point-à-point et de passage de messages collectifs

Bibliothèques de communication

2. MPI (Message Passing Interface)

Communicateur initial : MPI_COMM_WORLD

- **Existe comme le premier communicateur pour tous les processus dans l'application.**
- **Un ensemble de routines MPI existent pour former des communicateurs.**
- **Chaque processus à un "rang" dans un communicateur.**

Bibliothèques de communication

2. MPI (Message Passing Interface)

Exemple

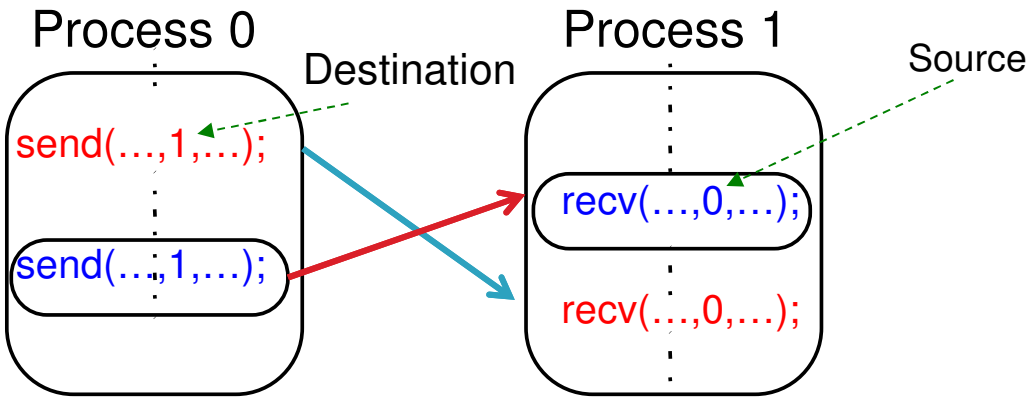
```
main (int argc, char *argv[])
{
        MPI_Init(&argc, &argv);
         /*find process rank */
        MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
        if (myrank == 0)            master();
                else       slave();
        MPI_Finalize();
}
```

Bibliothèques de communication

2. MPI (Message Passing Interface)

Risque

Comportement visé

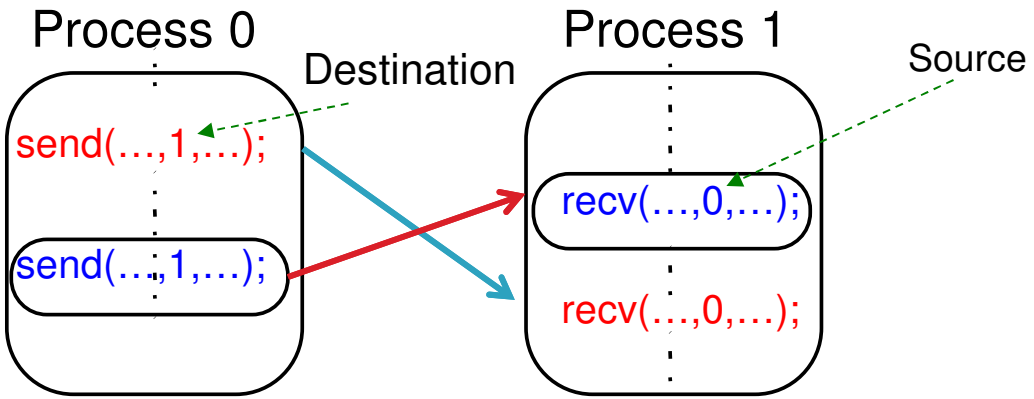


Bibliothèques de communication

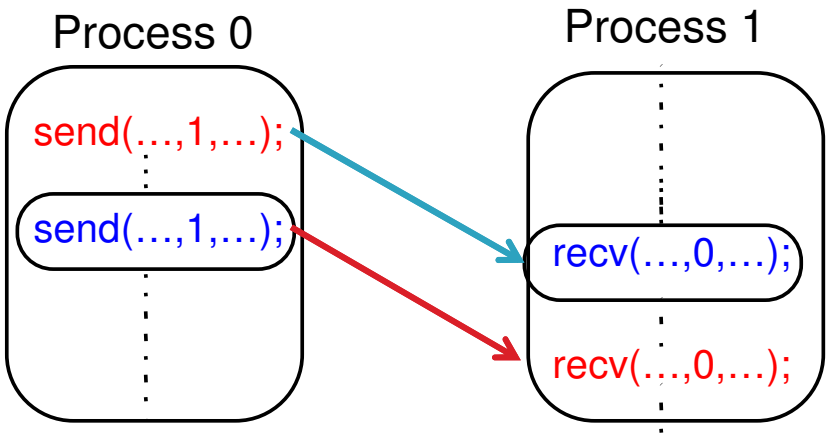
2. MPI (Message Passing Interface)

Risque

Comportement visé



Comportement possible



Bibliothèques de communication

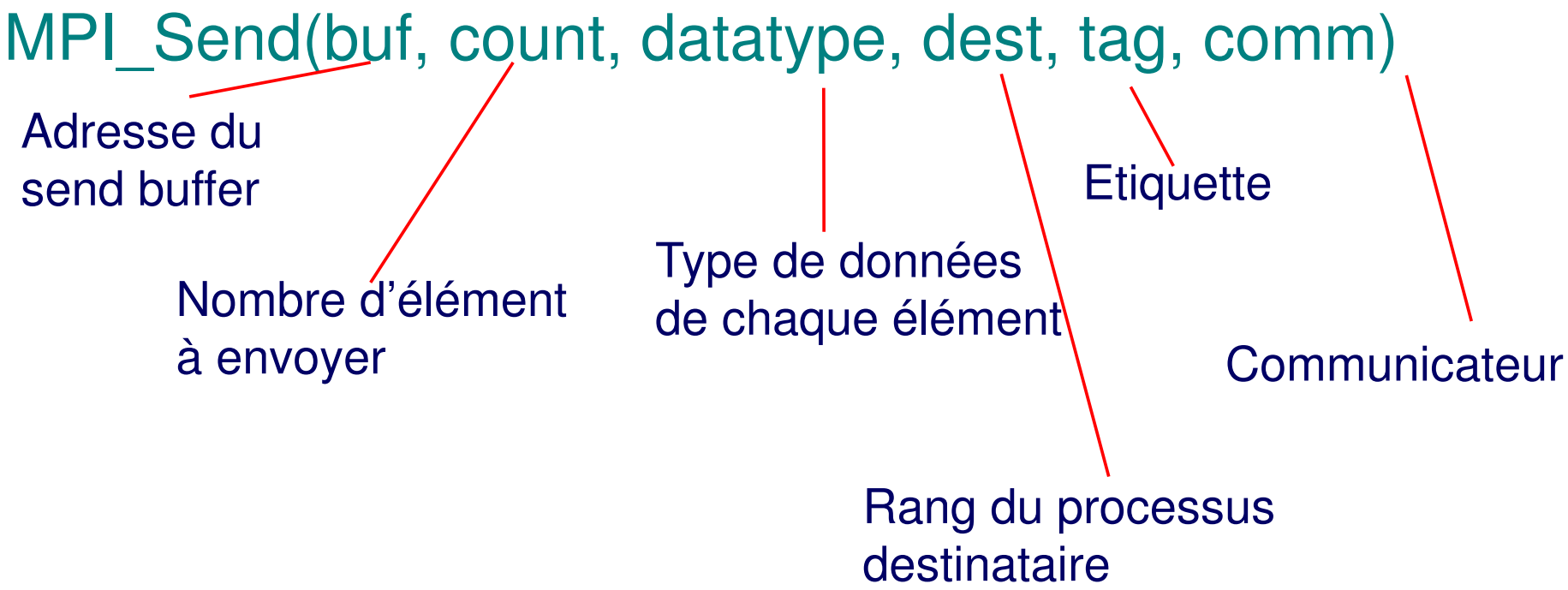
2. MPI (Message Passing Interface)

Solution : les étiquettes

- **Pour faire correspondance entre les opérations d'envoi et les opérations de réception.**

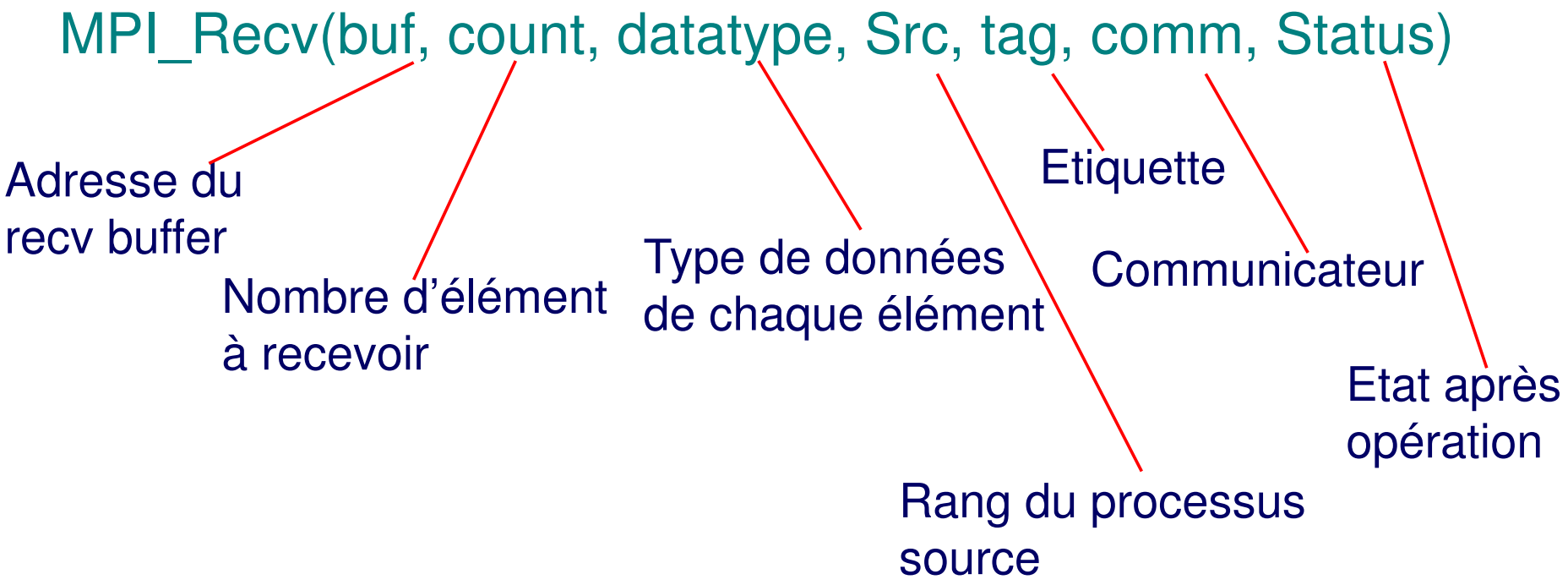
Bibliothèques de communication

2. MPI (Message Passing Interface)



Bibliothèques de communication

2. MPI (Message Passing Interface)



Bibliothèques de communication

2. MPI (Message Passing Interface)

Exemple : Pour envoyer un entier x du processus 0 au processus 1

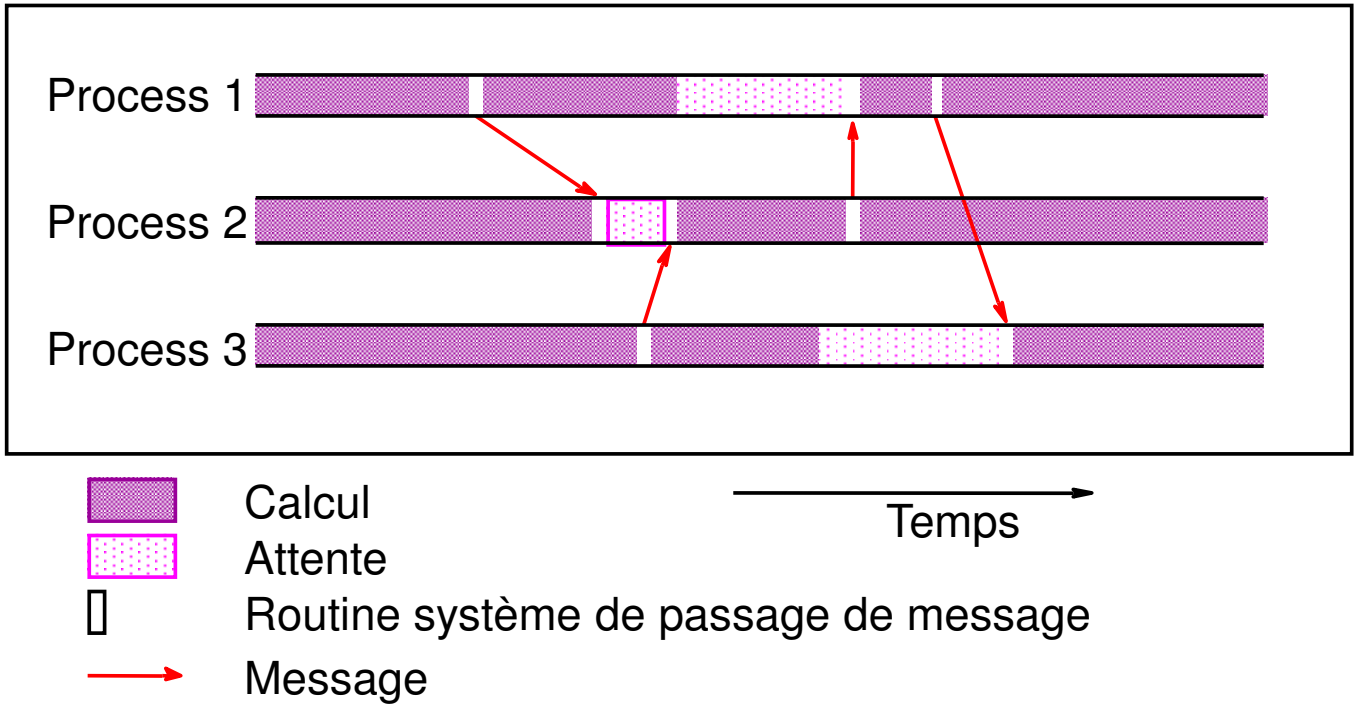
```
MPI_Comm_rank(MPI_COMM_WORLD, &myrank); /* trouve le rang */

if (myrank == 0)
 {
    int x;  x=20;
    MPI_Send(&x, 1, MPI_INT, 1, msgtag, MPI_COMM_WORLD);
 }  else if (myrank == 1)
{
    int x;
    MPI_Recv(&x, 1, MPI_INT, 0,msgtag,MPI_COMM_WORLD,status);
 }
```

Outils d'analyse de performance

1. Visualisation graphique

- **Les programmes peuvent être observés en cours d'exécution dans un diagramme espace-temps (ou diagramme de temps de traitement):**



Outils d'analyse de performance

1. Visualisation graphique

- Des implémentations d'outils de visualisation sont disponibles pour MPI.

- Un exemple est le "Upshot program visualization system".

Outils d'analyse de performance

2. Evaluation du temps

- Afin de mesurer le temps d'exécution entre deux points L_1 et L_2 dans un code, on peut avoir une structure comme suit:

```
L1: time(&t1);                                  /* start timer */
        .
        .
L2: time(&t2);                                  /* stop timer */
        .
elapsed_time = difftime(t2, t1);                /* elapsed_time = t2 - t1 */
printf("Elapsed time = %5.2f seconds", elapsed_time);
```

- MPI fournit la routine *MPI_Wtime()* pour retourner le temps (en secondes).