**Question 1  
a)**

Méthode 1 : Mémoire partagée

La mémoire partagée est la méthode la plus simple à utiliser. Elle permet à des processus d’accéder à la même zone mémoire.

Méthode 2 : Mémoire mappée

En ce qui concerne la mémoire mappée, elle ressemble à la mémoire partagée sauf que la communication s’effectue à l’aide d’un fichier.

Méthode 3 : Tube

Le principe des tubes est qu’ils sont unidirectionnels. Les données écrites sur l’extrémité d’écriture du tube sont lues depuis l’extrémité de lecture. Avec les tubes, les données sont toujours lues dans l’ordre où elles ont été écrites.

Méthode 4 : FIFO

La file FIFO est un type de tube. En effet, ce qui les différencie est que la file FIFO dispose d’un nom dans le système de fichier.

Méthode 5 : Sockets

Les sockets sont une communication de type bidirectionnel. Ils sont utilisés pour la communication entre processus sur une ou plusieurs machines.

**b)**

1) Pour la disponibilité du système. En effet si une machine est susceptible de devenir indisponible, il ne faudrait pas qu’un processus cesse de fonctionner à cause de cela. C’est pourquoi il est important de migrer un processus vers une autre machine afin de contrer le problème.

2) Pour la continuité des calculs. Prenons par exemple une personne qui se connecte à un réseau à l’aide d’un portable. Il pourrait vouloir transférer une application en cours d’exécution vers une autre machine sans pour autant arrêter l’application.

3) Pour le partage des ressources. Un processus pourrait ne pas disposer des ressources nécessaires sur la machine qu’il occupe en ce moment. C’est pourquoi il peut être pratique de migrer ce processus sur une machine disposant de plus de CPU disponible ou de mémoire et ainsi accélérer l’exécution du processus.

En résumé, les méthodes de communications entre processus sont diversifiées en 5 méthodes différentes qu’on pourrait voir en 3 types différents. La mémoire partagée et la mémoire mappée sont tous deux des méthodes qui partagent une ressource entre deux processus. Les tubes et les listes FIFO sont tous deux des méthodes de communication unidirectionnelle. Quant aux sockets, ils sont utilisés dans la programmation réseau, car les processus peuvent échanger des données entre plusieurs ordinateurs.

La mémoire partagée et la mémoire mappée sont les méthodes de communication entre processus les plus simples d’utilisation. C’est aussi les méthodes les plus rapides, car tous les processus partagent la même mémoire. Par contre, l’inconvénient est que ce partage provoque des conditions de concurrence. Cela se produit lorsque 2 processus ou plus veulent accéder en même temps à la mémoire. Pour pallier à ce problème, nous utilisons des sémaphores, des mutex et autres méthodes pour permettre au processus d’accéder à la mémoire chacun leur tour et non tous en même temps.

Les tubes et les listes FIFO sont un moyen de communication unidirectionnel. La différence entre les deux est que les listes FIFO disposent d’un nom dans le système de fichier et que les processus n’ont pas besoin d’avoir de lien de parenté pour pouvoir communiquer entre eux. Le principe unidirectionnel veut dire qu’à l’extrémité d’un tube on peut écrire et de l’autre lire. Cela a pour avantage que les données sont lues dans le l’ordre où elles ont été écrites. Cependant, les tubes et les listes ont une capacité limitée. Prenons par exemple un processus qui écrit plus vite que le processus qui li. Alors il y aura blocage pour le processus qui écrit et vice versa si un processus li plus vite que le processus qui écrit alors le processus qui li sera bloqué. Pour pallier ce problème, nous devons synchroniser les tubes et listes FIFO pour qu’il soit coordonné ensemble.

Pour finir, les sockets sont un type de communication bidirectionnel qui peut échanger des données sur un même ordinateur ou sur plusieurs. Ils ont l’avantage de consommer peu de ressources tout en étant très rapides. Par contre, les sockets ont comme inconvénient d’être plus difficiles à implémenter que les autres méthodes de communications.

En ce qui a trait au besoin de migrer un processus d’une machine à une autre, nous avons vu que pour plusieurs raisons il est préférable de pouvoir migrer un processus d’une machine à un autre. Tout d’abord pour anticiper une machine qui pourrait potentiellement s’arrêter au cours de l’exécution d’un processus, et cela en évaluant si un ordinateur est apte à exécuter un processus. Car si elle est susceptible de ne pas pouvoir exécuter un processus alors il est préférable de le transférer vers une machine qui elle peut. Par la suite, pour la continuité des calculs. En effet, il peut être important de continuer l’exécution d’un programme lors du transfert de celui-ci. Prenons par exemple notre cas numéro 1 ou une machine n’est plus en mesure de traiter un processus et qu’il l’a déjà démarré, mais que celui-ci ne peut pas être arrêté. Alors il est important de pouvoir migrer un processus entre machines tout en continuant l’exécution. Et pour finir, il est important de pouvoir migrer un processus d’une machine à une autre s’il se peut que celle-ci ne dispose pas des ressources nécessaires pour pouvoir exécuter le processus. Il peut être avantageux ici de pouvoir migrer un processus vers une machine disposant des ressources nécessaires pour pouvoir traiter le processus et ainsi s’assurer que le processus s’exécute correctement.