Université Larbi Ben M'hidi Oum El Bouaghi
Faculté des Sciences Exactes et des
Sciences de la Nature et de la Vie
Département de Mathématique et d'Informatique
Niveau : Master 2 Systèmes Distribués
Matière : Architectures Distribuées

Corrigé Type Contrôle 1

1. Dans une communication par message, comment assure-t-on la synchronisation entre un processus émetteur et un processeur récepteur ? On parle souvent de rendez-vous entre les deux processus, comment assure-t-on ce rendez-vous ? (2 pts).

- Dans une communication par message la synchronisation est assurée par les primitives *send* bloquant et *receive* bloquant. (1 pt)

  Lorsque les primitives *send* et *receive* sont les deux bloquants on parle de rendez-vous entre l'émetteur et le récepteur. (1 pt)

2. Une liaison logique entre processus communiquant est faite soit d'une manière directe ou indirecte. Détaillez chaque type de communication. (3.75 pts)

   a. Communication directe

   - Chaque processus voulant communiquer doit nommer explicitement le récepteur le récepteur ou l'émetteur de la communication. (0.25 pt)
     - ✓ Send(P, message) – Emettre un message vers P. (0.25 pt)
     - ✓ Receive(Q, message) – Recevoir un message de Q. (0.25 pt)
   - Dans ce schéma la liaison de communication a les propriétés suivantes :
     - ✓ Une liaison est établie automatiquement entre chaque paire de processus désirant communiquer. (0.25 pt)
     - ✓ Une liaison est associée à exactement deux processus. (0.25 pt)
     - ✓ Entre chaque paire de processus, il existe une seule et unique liaison. (0.25 pt)
   - Inconvénient de type de communication :
     - ✓ Le changement de l'identificateur d'un processus nécessite de revoir toutes les définitions du processus. (0.25 pt)

   b. Communication indirecte

   - Les messages sont envoyés et reçus par des boites aux lettres (appelées ports). Chaque boite aux lettres a un identificateur unique. (0.25 pt)

- Dans ce schéma le processus peut communiquer avec d'autres processus via un nombre de boites aux lettres. (0.25 pt)
- Deux processus ne peuvent communiquer que s'ils ont une boite aux lettres partagées. (0.25 pt)
  - ✓ Send(A, message)– Envoyer un message vers la boite aux lettres A. (0.25 pt)
  - ✓ Receive(A, message) – Recevoir un message de la boite aux lettres A. (0.25 pt)
- La liaison de communication a les propriétés suivantes :
  - ✓ La liaison est établie entre une paire de processus seulement si les deux processus membres de la paire ont une boite aux lettres partagée. (0.25 pt)
  - ✓ La liaison peut être associée à plus de deux processus. (0.25 pt)
  - ✓ Entre chaque paire de processus qui communiquent, il peut exister plusieurs liaisons différentes, chacune d'entre elles a une boite aux lettres. (0.25 pt)

3. On parle généralement de l'existence de plusieurs de niveaux de transparence, quels sont ces niveaux ? tout en détaillant chaque niveau. Quels sont les niveaux de transparence qui sont considérés comme une transparence de réseau? Quel est l'effet de l'absence de cette transparence ? (7 pts)

- Transparence d'accès :
  - La même interface d'accès que ce soit pour une ressource locale ou une ressource distante. (0.25 pt)
  - La syntaxe utilisée ne doit pas être différente. (0.25 pt)
  - L'accès aux données indépendamment de leurs formats de représentation. (0.25 pt)

- Transparence de localisation :
  - La désignation de la ressource est indépendante de sa localisation. (0.25 pt)
  - La nécessité d'un service de nommage global (annuaire). (0.25 pt)

- Transparence de concurrence (partage) :
  - Les accès concurrents à la ressource sont contrôlés de telle façon que l'intégrité de la ressource soit assurée. (0.25 pt)
  - La synchronisation des accès en lecture et en écriture à la ressource. (0.25 pt)

- Transparence de réplication (duplication) :
  - Consiste à s'assurer que l'accès à une ressource soit identique quel que soit la forme d'implantation d'une ressource répliquée. (0.25 pt)
  - Utilisé pour augmenter la fiabilité et la performance. (0.25 pt)
  - L'accès à une ressource répliquée est indépendant de l'indisponibilité d'une de ses occurrences. (0.25 pt)

- Transparence de panne :
  - Détection de la défaillance des nœuds du système distribué. (0.25 pt)

  - Pallier à cette défaillance d'une manière transparente. (0.25 pt)
  - Ne pas bloquer le fonctionnement global de l'application. (0.25 pt)
  - La réplication permet une transparence de l'indisponibilité d'une ressource et/ou d'un service. (0.25 pt)

- Transparence de mobilité (migration) :
  - La propriété de migration consiste à assurer qu'une ressource peut migrer d'un nœud à un autre sans que les usagers s'en aperçoivent. (0.25 pt)
  - Déplacer un service dynamiquement vers un serveur moins chargé. (0.25 pt)
  - Mettre en œuvre des stratégies de régulation de charge sur une architecture distribuée. (0.25 pt)

- Transparence de performance (charge) :
  - La régulation de la charge de chaque nœud peut permettre une meilleure exploitation du système global et une meilleure satisfaction des utilisateurs. (0.25 pt)
  - La mise en œuvre est délicate car une régulation de la charge globale implique une bonne connaissance de l'état global du système (difficile à obtenir). (0.25 pt)
  - Nécessite la mise en place d'algorithmes répartis spécifiques. (0.25 pt)
  - Nécessite une bonne connaissance des contraintes du réseau utilisé. (0.25 pt)

- Transparence d'échelle :
  - Un changement d'échelle d'une application distribuée consiste à augmenter le nombre de nœuds. (0.25 pt)
  - Ceci est facilité par la propriété modulaire et dynamique d'une architecture distribuée. (0.25 pt)
  - L'ajout de nœuds ne doit pas nécessiter l'arrêt du système. (0.25 pt)
  - Ce changement d'échelle n'est pas forcement transparent pour les usagers, il faut donc le contrôler et le rendre le plus transparent possible. (0.25 pt)

- Les deux plus importantes transparences sont l'accès et la localisation considérées comme transparence de réseau. (0.5 pt)

- La présence ou l'absence de la transparence de réseau affecte fortement l'utilisation des ressources distribuées. (0.25 pt)

4. Dans une application client/serveur, on distingue trois niveaux, détaillez chaque niveau. Comment peut-on représenter ces trois niveaux dans une architecture 2-tiers ? (4.5 pts)

Dans une application client/serveur, on distingue trois niveaux :

- Niveau interface utilisateur (Présentation) : (0.25 pt)
  - Partie cliente d'une application implémente l'interface utilisateur. (0.25 pt)
  - Ce niveau gère l'interaction de l'utilisateur avec l'application (formule les requêtes, affichage graphique, etc.). (0.25 pt)

- Niveau gestion de données : (0.25 pt)
  - Partie de l'application contenant les programmes qui maintiennent ses données. (0.25 pt)
  - Le niveau de données est matérialisé par (des fichiers, bases de données). (0.25 pt)
  - Cas général les données sont du coté serveur. (0.25 pt)

- Niveau traitement (logique métier) : (0.25 pt)
  - Dépend des applications. (0.25 pt)
  - Ensemble de fonctionnalités situé entre le niveau de données et le niveau interface. (0.25 pt)

Ces trois niveaux sont représentés dans l'architecture 2-tiers de la manière suivante :

- Client/serveur de présentation : (0.25 pt)
  - Type 1 (Présentation distribuée) : (0.25 pt)
  - Type 2 (Présentation distante) : (0.25 pt)
- Client/serveur de Traitements : (0.25 pt)
  - Type 3 (traitement distribué) : (0.25 pt)
- Client/serveur de données : (0.25 pt)
  - Type 4 (gestion distante de données) : (0.25 pt)
  - Type 5 (Bases de données distribuées) : (0.25 pt)

5. Quels sont les différents types de serveurs qui existent tout en détaillant le fonctionnement de chaque type. (3 pts)

   - Il existe deux types de serveurs: serveurs itératifs et serveurs concurrents :
     - Serveur Itératif: itère sur les étapes suivantes : (0.25 pt)
       - ✓ Attendre l'arrivée d'une requête du client. (0.25 pt)
       - ✓ Traiter la demande et envoyer une réponse au client. (0.25 pt)
       - ✓ Revenir à l'étape d'attente. (0.25 pt)
     - Le serveur itératif traite les requêtes des clients séquentiellement, finir avec un client avant de servir le prochain. (0.25 pt)
     - Serveur concurrent : (0.25 pt)
       - ✓ Attendre l'arrivée d'une requête du client. (0.25 pt)
       - ✓ Utiliser un nouveau processus/thread pour traiter la requête. (0.25 pt)
       - ✓ Revenir à l'étape d'attente. (0.25 pt)
     - Le serveur concurrent traite les requêtes des clients parallèlement. (0.25 pt)
     - Il existe deux approches pour implanter le serveur concurrent :
       - ✓ Thread par client : un nouveau thread est affecté à chaque requête. (0.25 pt)
       - ✓ Pool de threads : un ensemble de threads réutilisables à tour de rôle selon la stratégie round-robin pour traiter les requêtes. (0.25 pt)