Université Larbi Ben M'Hidi
Faculté des sciences exactes et des sciences de la nature et de la vie
Département M.I. - 1ière Année Master Architectures Distribuées

18 Jan. 2023
Dr. Kouah S.

Contrôle d'Ingénierie des Logiciels Distribués

Questions de Cours (5 Points)

1. Donner trois caractéristiques des systèmes distribués.
2. Qu'elle est la différence entre la logique temporelle linéaire et la logique temporelle arborescente ?
3. Qu'est-ce qu'un système distribué ?
4. **Répondre par vrai ou faux :**
   a) L'architecture Peer to Peer centralisée est une architecture centralisée.
   b) Dans une architecture Peer to Peer, tous les processus jouent des rôles différents (clients ou serveurs).
   c) L'architecture client/serveur à deux tiers est une architecture centralisée.
   d) Il n'y a aucune différence entre les opérateurs «>> » et « ; » de LOTOS.
   e) Un système distribué est un système ouvert, hétérogène et transparent, dont toutes les horloges sont parfaitement synchronisées.
   f) Les automates de Büchi sont des cas particuliers des automates de Muller, mais ils sont déterministes.
   g) Les sockets et le RMI sont des middlewares assurant la propriété d'hétérogénéité.

Exercice 1 (5 Points)

On veut modéliser le comportement d'un **ascenseur** lors d'un **appel**.

1. Un ascenseur peut être modélisé par un **automate** à **deux variables** (**X** : étage de l'ascenseur et **Y** : étage appelé) **et trois états** (**S0** : arrêté, **S1** : montant et **S2** : descendant). Construire un tel automate.
2. Ecrire la **propriété** suivante en utilisant la logique **temporelle linéaire** : « Toujours un appel d'ascenseur finira par une réponse ».

Exercice 2 (6 pts)

Répondre à une seule question (1 ou 2) :

1. Spécifier le **problème de l'exercice précédent** en utilisant le langage de spécification formelle **LOTOS**.
2. Décrire, en **LOTOS**, les spécifications formelles de trois processus parmi les processus communicants suivants :
   a) **Producteur-Consommateur** dans lequel un processus **PRODUCER** envoie des informations à un processus **CONSUMER** via un processus tampon **BUFFER**.
   b) **Client-Serveur**, où plusieurs processus **Clients** sont en concurrence pour l'accès à une ressource **G** fournie par un processus **SERVER** :
   c) **Réseau en étoile**, où plusieurs processus sites communiquent par l'intermédiaire d'un processus central appelé **NODE**.
   d) **Réseau en anneau**, où chaque processus site ne peut communiquer qu'avec ses voisins immédiats.

Exercice 3 (4 pts)

1. Expliquer le rôle des classes java suivantes : **InetAddress**, **Thread**, **ServerSocket**, et **Socket**.
2. Expliquer en bref, comment créer une application à base de **RMI** ?
3. Comment créer les **threads** en java, expliquer tous les cas possibles.

Bon courage

Université Larbi Ben M'Hidi
Faculté des sciences exactes et des sciences de la nature et de la vie
Département M.I. - 1ière Année Master Architectures Distribuées

18 Jan. 2023

Dr. Kouah S.

Corrigé Type du contrôle Ingénierie des Logiciels Distribués

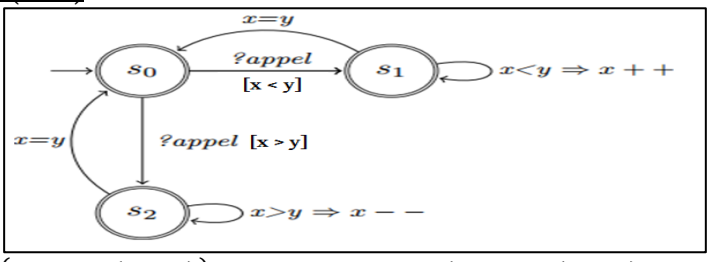
Questions de Cours (6 Points)

1. Trois caractéristiques des systèmes distribués : **Transparence**, **Hétérogénéité**, **Ouverture**, etc. **(0.5 pts × 3)**
2. La logique temporelle linéaire utilise une **vision linéaire de temps ;** cependant, la logique temporelle arborescente utilise une **vision arborescente de temps (0.75 pts).** D'autres réponses portant sur la syntaxe sont acceptées.
3. **Un système distribué** : est une collection d'ordinateurs indépendants qui apparaissent à l'utilisateur comme un seul système cohérent. **(0.75 pts)**
4. **Répondre par vrai ou faux :**

a) Faux, L'architecture Peer to Peer centralisée est une architecture **Distribuée**. **(0.5 pts)**
b) Faux, Dans une architecture Peer to Peer, tous les processus jouent des **rôles similaires** et coopèrent égal a égal pour réaliser une activité repartie. **(0.5 pts)**
c) Faux, L'architecture client/serveur à deux tiers est une architecture **distribuée**. **(0.25 pts)**
d) Faux, l'opérateur «>> » **représente une composition séquentielle entre processus** ; cependant, l'opérateur « ; » **représente un pré fixage par action**, **(0.5 pts)**
e) Faux, les horloges **ne sont synchronisées**. **(0.25 pts)**
f) Faux, **c'est l'inverse**, les automates de Muller sont des cas particuliers des automates de Büchi, mais ils sont déterministes. **(0.5pts)**
g) Faux, les sockets **ne sont pas un middleware**. **(0.5 pts)**

Exercice 1 (5 Points)

1. **Spécification par automate (3 Pts)**



2. **Propriété LTL (2 Pts)** : $G\ ((appel\ \wedge (x < y)) \rightarrow F(x = y))\ \vee\ G\ ((appel\ \wedge (x > y)) \rightarrow\ F(x = y))$.
   Ou bien : $G\ (appel \rightarrow F(x = y))$ tel que $(x = y)$, spécifie « l'ascenseur se trouve à l'étage demandé » (appel répondu).

Exercice 2 (6 Pts):

1. **Spécification en LOTOS**

Specification Ascenseur [appel]: **noexit**
Behavior
appel ?Y ;
([X<Y] →Q **) [ ] (** [X>Y] →Q')
Where Process Q : **noexit** :=
([X<Y] → (X++ ; Q)) [ ] ([X=Y] → exit)
Endproc
Process Q' : **noexit** :=
([X>Y] → (X - - ; Q')) [ ] ([X=Y] → exit)
Endproc
Endspec

2. **Il suffit de donner le comportement, comme suit : (2 pts × 3)**

a) PRODUCER [G1] ||[G1]| BUFFER [G1, G2] ||[G2]| CONSUMER [G2].
b) (CLIENT [G] ||| CLIENT [G] ||| ... ||| CLIENT [G]) ||[G]| SERVER [G].
c) (SITE [G1] ||| SITE [G2] ||| ... ||| SITE [Gn]) ||[G1, G2, ..., Gn]| NODE [G1, G2, ..., Gn].
d) (SITE [G0, G1] ||[G1]| SITE [G1, G2] ||[G2]| ... ||[Gn-1]| SITE [Gn-1, Gn]) ||[Gn, G0]| SITE [Gn, G0].

Exercice 3 (4 Points):

1. **(0.5 pts × 4)**

- InetAddress : Cette classe représente les adresses IP et un ensemble de méthodes pour les manipuler.
- Thread : permet de créer et manipuler les threads, qui permettent une exécution concurrente de taches indépendantes dans un programme java.
- ServerSocket : permet de créer une socket coté serveur, écoutant les connexions entrantes, permettant ainsi la communication avec des clients.
- Socket : permet de créer des sockets coté client, facilitant l'établissement de connexions réseau bidirectionnelle entre clients et serveurs.

2. Création d'une application à base de **RMI : (0.25 pts × 4)**
   - Définir une interface distante,
   - Implémenter cette interface dans une classe serveur,
   - Créer un registre RMI,
   - Puis développer une classe cliente qui utilise la référence distante (objet distant) pour invoquer les méthodes du serveur.
3. Création des **threads** en java : on peut **étendre la classe Thread** ou **implémenter l'interface Runnable**, puis instancier et démarrer le thread. **(0.5 pts × 2)**