Oral de rattrapage

Exercice 1 (4 points)

Voici quelques questions de réflexion et rédaction en relation avec le cours

(a) (2 points) Sous quelles hypothèses peut on se limiter à raisonner en terme de probabilité d'occurrence de défaillances lorsque l'on souhaite raisonner sur la vraisemblance des incidents. Donnez l'exemple d'un type d'incident qui ne peut pas se produire dans ce cas.

(b) (1 point) Expliquez pourquoi il est possible d'utiliser la réplication active lorsque la réplication passive est possible mais pas l'inverse (la réponse doit être justifiée) ?

(c) Quel est l'objectif principal de la méthode HAZOP dans un processus d'analyse de risque (que feriez vous de son résultat ?)

Exercice 2 (3 points)

Nous souhaitons étudier l'implémentation d'une fonction de calcul de plus grand commun diviseur (pgcd) du point de vue de la Tolérance aux Fautes. Pour commencer, nous allons identifier les incidents/défaillances pouvant toucher ce système. Pour cela, il est nécessaire d'avoir une spécification de cette fonction de calcul qui retourne le pgcd de deux entiers :

Définition 1 : La fonction de calcul du plus grand commun diviseur ou PGCD de deux nombres entiers a et b retourne le plus grand entier inférieur à a et b qui les divise tous les deux.

Rappel : on supposera en toute théorie qu'il n'y a pas de diviseur de 0.

Une implémentation de cette spécification est fournie en C.

```
1  long find_gcd(long x, long y)
2  {
3      if(x > y)
4         { return find_gcd(x-y, y);}
5      else {
6          if(y > x)
7          { return find_gcd(x, y-x);}
8          else return x;
9          }
10 }
```

(a) ($\frac{1}{2}$ point) Expliquez pourquoi l'on peut dire que cette implémentation engendre un incident sur l'état de contrôle du processus dans lequel elle serait exécutée si on l'appelle sur le couple de paramètre (0,4).

(b) (1 point) Expliquez pourquoi la définition 1 est une mauvaise spécification du point de vue sûreté de fonctionnement. Cela aurait-il été différent en exigeant a et b entiers naturels ?

Exercice 3 ($3\frac{1}{2}$ points)

Nous définissons dans le cours le concept de zone de confinement d'erreur comme un composant (logiciel) capable de détecter ou empêcher la propagation des états d'erreurs entre l'environnement du composant et son état interne (i.e. état non présent sur l'interface).

Nous considérons à nouveau l'implémentation de la fonction de calcul de pgcd.

```
1  long find_gcd(long x, long y)
2  {
3      if(x > y)
4         { return find_gcd(x-y, y);}
```

```
5       else {
6           if(y > x)
7           { return find_gcd(x, y-x);}
8           else return x;
9           }
10  }
```

La spécification de cette fonction est désormais :

Définition 2 : La fonction de calcul du plus grand commun diviseur ou PGCD prend en entrée deux nombres entiers naturels positifs a et b, et retourne le plus grand entier naturel positif qui les divise tous les deux.

(a) (1 point) Appliquez le principe de l'enveloppe pour vous assurer que l'on ne peut activer de faute d'interaction pour cette fonction entraînant une défaillance de la fonction (on supposera que pour tout couple d'entiers dont la somme est inférieure à N, la récursion n'engendre pas de débordement de pile).

(b) ($\frac{1}{2}$ point) Est ce que l'on peut obtenir un comportement similaire à celui de la fonction enveloppée et les mêmes garanties en utilisant simplement un type primitif différent pour les paramètres d'entrée pour le langage C ? (jusitifiez)

Exercice 4 (2 points)

Nous souhaitons déployer une architecture de réplication active qui repose à l'état initial sur N répliques actives pouvant subir des fautes byzantines. Les calculs à l'échelle du système doivent être réalisés de manière périodique. Les entrées sont envoyées aux répliques avec une période T, le résultat doit être produit par le voteur avant l'arrivée d'une nouvelle donnée sur les répliques.

Chaque réplique possède un temps d'exécution borné par une échéance relative de $2 * T/3$ dans le cas où la réplique n'est pas défaillante. Au début de chaque période toutes les répliques reçoivent une nouvelle donnée (pas de délai ici). On suppose les temps de transmission entre réplique et voteur très inférieurs devant $T/6$ donc modélisés comme nuls.

(a) (1/2 point) La réplication active consiste à collecter les résultats des répliques (attendre qu'ils soient transmis au voteur) puis à produire la valeur majoritaire. Décrivez une solution sur cette architecture utilisant le nombre minimal de répliques pour tolérer au maximum 2 défaillances byzatines.

On souhaite que le voteur donne une indication sur le niveau de confiance que l'on peut placer dans la valeur qu'il produit.

(b) (1/2 point) Combien de défaillances le système subit le plus vraisemblablement si l'on a observé 2 valeurs distinctes une valeur de fréquence 4 et une valeur de fréquence 1. Justifiez votre réponse (On fait l'hypothèse que les défaillances sont indépendantes). (Il pourra être nécessaire de faire une hypothèse sur les comportements aléatoires considérés).

(c) (1 point) Expliquez pourquoi du point de vue du voteur, le cas où il obtient seulement deux résultats différents mais tous les deux avec une fréquence de 2 est pire que d'avoir 1 résultat avec fréquence de 3 et un résultat avec une fréquence de 2 (ici pire signifie correspond a priori à un plus grand nombre de défaillance).

Exercice 5 (2 points)

ON implémente le voteur d'une réplication active d'une fonction périodique avec une unique file d'attente pour collecter les réponse en l'absence de moyen de prouver l'origine des messages. La file est vidée à chaque période.

(a) Expliquez pourquoi une unique réplique byzzantine suffit à déclencher une défaillance dans ce cas

(b) Supposons maintenant que l'on identifie les émetteur que doit faire le voteur avant de calculer les fréquences.