SCIENCES SORBONNE UNIVERSITÉ

TME ARA
PeerSim : Algorithmes de diffusion

Jonathan Lejeune

Objectifs

L'objectif de ce TME est d'implanter les différents algorithmes de diffusion vus en cours. Il vous aidera à comprendre les différences entre les classes de diffusion (fiable, totale, causale, ...) tout en utilisant au maximum les fonctionnalités de PeerSim.

Exercice 1 – Diffusion, fiabilité et cohérence

Question 1

Créez dans votre projet Eclipse copiez-y dans le dossier source correspondant les fichiers fournis dans vos ressources TME.

Question 2

Il est également nécessaire d'avoir accès à la classe Message dont la code a été été fourni au TME de prise en main. Pour ce faire deux solutions s'offrent à vous :

— soit vous copiez-collez la classe message dans votre projet (mais redondance de code),

— soit vous référencez votre projet eclipse du TME de prise en main dans le build path du projet de diffusion (clic droit sur le projet → *Build Path* → *Configure Build Path* → onglet *Projects* → bouton *Add...*).

Question 3

Rafraîchissez et assurez-vous que la compilation se passe bien (absence de croix rouge).

Les fichiers qui vous ont été fournis implémentent un protocole applicatif (ApplicativeProtocol.java) qui se base sur des primitives de diffusion. Ce protocole est une succession de diffusions parmi N processus. Chaque processus maintient le réplica d'une variable entière initialisée à 0 et font chacun deux diffusions de modifications : une addition suivie d'une multiplication. La valeur ajoutée ou multipliée est l'ID du node + 1 (pour éviter d'avoir 0 pour le processus 0). Ces diffusions ont un ordre causal : un processus émet ses modifications si il a reçu l'opération de multiplication de processus ayant le premier identifiant inférieur non fautif. Le but de l'exercice est d'empiler différents protocole de diffusion afin d'assurer qu'à termes tous les nœud non fautifs aient la même valeur.

Question 4

Analysez le code qui vous a été fourni et produisez un fichier de configuration peersim de manière à :

- d'avoir une graine aléatoire à la valeur 5
- ce que le protocole applicatif fonctionne sur 5 nœuds en se basant sur le protocol de diffusion basique (BasicBroadcast.java)
- utiliser une couche transport fiable (prendre UniformRandomTransport de l'API peersim) et d'avoir une latence réseau comprise entre 5 et 500 unités de temps
- que le module d'initialisation soit la classe Initialisation
- que le contrôleur EndControler soit exécuté une fois la simulation terminée
- que le temps de simulation soit suffisamment grand pour que le protocole applicatif se termine

Notez les résultats.

Question 5

Testez la même expérience en simulant des pannes franches de nœud à l'émission. Pour cela utilisez la classe DeadlyTransport en parametrant que les nœuds 1 et 3 peuvent tomber en panne avec une probabilité de 0.1. Notez les résultats vérifiez que le broadcast ne respecte pas la spécification d'un broadcast fiable.

Question 6

Testez la même expérience en appliquant cette fois ci le protocole de diffusion fiable (ReliableBroadcast dont l'algorithme est donné en cours) sur le protocole de transport non fiable. Notez les résultats et vérifiez que la spécification du broadcast fiable est respectée.

Question 7

En respectant les même mécanismes que le code qui vous a été fourni

- codez une classe de broadcast (implémentant l'interface broadcast) qui implémente l'algorithme de diffusion FIFO présenté en cours. Cette classe se basera sur un protocole de broadcast qui sera spécifié dans le fichier de configuration.
- Testez votre implémentation dans un milieu fiable (pas de pannes franches)
- Vérifiez que les affichage correspondent bien à une diffusion FIFO.
- Notez les résultats.

Question 8

Même question mais en implémentant une diffusion causale qui se basera sur un protocole de diffusion FIFO. Notez les résultats

Question 9

Créez une classe de broadcast qui implémente l'algorithme de diffusion total à séquenceur fixe. Testez cette classe en vous basant sur un transport fiable et le broadcast

basique et vérifiez que les valeurs de chaque réplicas sont égales. L'identifiant du séquenceur pourra être paramétrable dans le fichier de configuration.

Question 10

Testez votre code avec plusieurs combinaisons de protocoles :

— reliableTransport + reliablebroadcast + fifobroadcast + applicativeprotocol
— reliableTransport + reliablebroadcast + fifobroadcast + causalbroadcast + applicativeprotocol
— reliableTransport + reliablebroadcast + totalbroadcast + applicativeprotocol
— reliableTransport + reliablebroadcast + totalbroadcast + fifo broadcast + applicativeprotocol
— reliableTransport + reliablebroadcast + totalbroadcast + fifo broadcast + causalbroadcast + applicativeprotocol

Testez également avec DeadlyTransport à placer entre reliableTransport et reliablebroadcast