UPMC
SORBONNE UNIVERSITÉS

TME ARA 2017–2018
PeerSim : Prise en main et Algorithmes de diffusion

Jonathan Lejeune

Objectifs

L'objectif de ce TME est de vous familiariser avec l'outil PeerSim en implémentant 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 – Installation de l'environnement de travail

Question 1

Téléchargez le simulateur PeerSim et désarchivez-le :

```
wget http://downloads.sourceforge.net/project/peersim/peersim-1.0.5.zip

unzip peersim-1.0.5.zip
```

Question 2

Ouvrez Eclipse et créez un nouveau projet Java (nommons-le par exemple TMEPeerSim).

Question 3

Pour compiler et exécuter vos programmes PeerSim dans Eclipse, vous devez ajouter au BuildPath de votre projet les jars les différents fichiers jar présents à la racine de votre dossier d'installation PeerSim. Pour cela :

- Sélectionnez *Build Path* dans le menu contextuel (clic droit sur le dossier) du projet puis *Configure Build Path*
- Dans l'onglet *Libraries*, cliquez sur *Add External JARs....*
- Ajoutez l'ensemble des *Jar* présents à la racine de votre dossier Peersim
- Vérifiez que les Jar ont bien été ajoutés dans le répertoire *Referenced Libraries* dans votre projet Eclipse.

Question 4

Afin de pouvoir parcourir le code source de PeerSim dans Eclipse (ceci peut être utile pour comprendre davantage son fonctionnement) :

— Zippez le dossier src de votre installation
`zip -r src.zip src`

— Dans Eclipse, liez cette archive de fichiers sources au jar peersim-1.0.5.jar. Pour ceci :

  — Clic droit sur le fichier dans les *Referenced Libraries* puis *Properties*

  — Dans le menu *Java Source Attachment* sélectionnez *External Location* et indiquez l'archive zip en cliquant sur le bouton *External File*

Exercice 2 – Diffusion, fiabilité et cohérence

Question 1

Créez dans votre projet Eclipse un package *tme1* et copiez-y dans le dossier correspondant les fichiers fournis dans vos ressources TME. Rafraîchissez et assurez-vous que la compilation se passe bien (absence de croix rouge).

Question 2

Les fichiers qui vous ont été fournis implémentent un protocole applicatif (ApplicativeProtocol.java) qui se base sur des primitives de diffusion. Déterminez ce que fait ce protocole.

Question 3

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 4

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 5

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 6

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 7

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

Question 8

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 9

Testez votre code avec avec plusieurs combinaisons de protocole :

— 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