UNIVERSITÉ DE MONS-HAINAUT

FACULTÉ DES SCIENCES

SIMULATION DE SYSTÈMES À ÉVÈNEMENTS DISCRETS

Rapport du projet

Auteurs :
Sébastien DUBOIS
Jean-François MERNIER
Frédéric REGNIER

4 mai 2009

Table des matières

Table des figures

1 Evenements

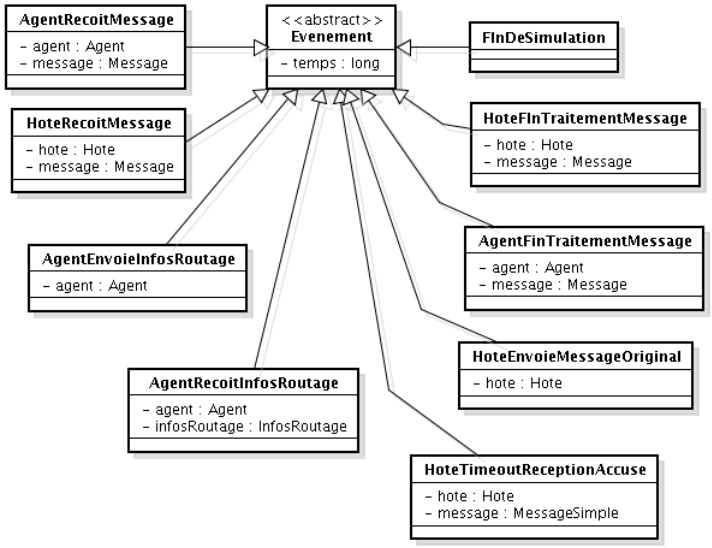

FIG. 1 – Evènements

1.1 Hotes

1.1.1 Envoi d'un message original

1.1.2 Réception d'un message

1.1.3 Fin de traitement d'un message

1.1.4 Timeout

1.2 Agents

1.2.1 Réception d'un message

Dans le diagramme UML, il y a deux points que nous n'avons pas expliqué. Tout d'abord : « Est-ce qu'on peut envoyer les nouvelles informations de routage maintenant ? ». En fait nous avons décidé d'éviter trop d'envois successifs inutiles d'informations de routage, afin de ne pas surcharger le système. Pour ce faire, quand on doit envoyer les informations de routage (à cause du niveau d'occupation du buffer), on vérifie si ça fait au moins x temps de simulation qu'on a envoyé un message. Si oui, alors on peut envoyer le message. De cette manière, si pour un temps t donné, l'agent reçoit 50 messages, que pour le premier il dépasse le seuil d'alerte d'occupation du buffer et envoie ses informations de routage, puisqu'il sera toujours au delà du seuil d'alerte pour les 49 autres messages, on évite ainsi de réenvoyer les informations inutilement.

Le second point : « Augmentation du coût de nos routes à destination des autres agents (on augmente d'une valeur fixe à chaque fois) »est ce que le distance vector prenne en compte le niveau d'occupation des agents. Quand un agent donné est surchargé, il augmente le coût de ses routes à destination d'autres agents et prévient ses voisins. De cette manière quand

les voisins reçoivent les informations, ils mettent à jour leur propre DV et choisissent peut être d'autres routes pour faire suivre les messages (sauf si l'agent est la destination finale bien sûr !). Si les autres agents deviennent surchargés, leurs coûts augmenteront également. Ainsi au final, le Dv prend en compte l'occupation des buffers des agents, ce qui permet de mieux répartir la charge sur les différents agents. Dans les résultats des simulations, nous avons observé une bien meilleure répartition.

1.2.2 Fin de traitement d'un message

1.2.3 Envoi des informations de routage

1.2.4 Réception d'informations de routage

2 Résultats

2.1 Paramètres du système

2.1.1 Hote

- Durée du timeout (temps après lequel on réémet un message)
- Temps maximal inter-envois (pour les messages originaux)
- Temps de traitement d'un message
- Pourcentage de messages à destination d'un autre agent

2.1.2 Agent

- Nombre d'hôtes reliés
- Taux de pertes brutales de messages
- Temps de traitement d'un message
- Taille de buffer (en entrée)

Pour le distance vector on a en plus :

- Temps inter-envois des informations de routage

2.1.3 Simulation

- Durée
- Délai agent <-> hôte
- Distance vector activé (oui/non)
- Durée de la période d'initialisation
- Périodicité d'affichage des statistiques (e.g., tous les 1% de simulation)

3 Décisions

3.1 Gestion des évènements

Nous avons choisi d'utiliser une seule FEL pour la simulation. On y place tous les évènements. De plus, pour un temps t de simulation donné, nous avons décidé de traiter certains évènements prioritairement :

1. En premier lieu on traite les évènements de réception d'informations de routage
2. Puis les évènements de réception d'informations de routage
3. Puis les évènements de réception de messages (accusés et messages normaux)
4. Puis les évènements de timeout (un message pour lequel on a pas encore reçu d'accusé)

Une fois tous ces évènements traités pour un temps t, on traite les autres selon l'ordre *FIFO*.

3.2 Distance vector

Nous avons choisi de modifier les coûts en fonction du taux d'occupation du buffer de l'agent. Ceci est expliqué à la section 1.2.1 (p5) concernant l'évènement de réception d'un message par un agent.

TODO continuer

A Le programme et son utilisation

L'exécutable du projet est disponible dans le dossier target/dist. Pour l'exécuter, il suffit d'ouvrir un prompt et de lancer : **java -jar simulation.jar**.

A.1 Configuration et aide

Pour afficher la liste des paramètres pouvant être donnés au programme, il suffit d'ouvrir un prompt et de lancer : **java -jar simulation.jar –aide**

```
Option                                  Description
------                                  -----------
-a, -h, --aide, --help                  Aide
--agentsNombreHotes <Long>              Nombre d'hotes par agent
--agentsTailleMaxBuffer <Long>          Taille des buffers des agents (>= 0)
--agentsTauxPerteBrutale <Float>        Taux de perte brutale des agents (e.
                                          g., 0.05) (0 <= valeur < 1)
--agentsTempsInterEnvoiInfosRoutage     Temps entre deux envois des
  <Integer>                               informations de routage (>=0)
--agentsTempsTraitementMessage <Float>  Temps de traitement d'un message par
                                          un agent (0 <= temps traitement <=
                                          1). 0 = traitement instantané
--delaiEntreEntites <Integer>           Délai nécessaire pour qu'un message
                                          d'un hôte arrive à l'agent (et
                                          inversément) (>=0)
--duree <Long>                          Durée de la simulation (> 0)
--dureeInitialisation <Long>            Durée de la période d'initialisation
                                          de la simulation (>= 0)
--dvActive                              Pour activer le distance vector (si
                                          non spécifié, désactivé!)
--hotesTauxMessagesVersAutreAgent       Taux de messages d'un hôte qui seront
  <Float>                                 à destination d'un hôte relié à un
                                          autre agent (e.g., 0.75) (0 <=
                                          valeur <= 1)
--hotesTempsMaxInterEnvois <Integer>    Temps maximal entre deux envois d'un
                                          hôte (> 0)
--hotesTempsTraitementMessage <Float>   Temps de traitement d'un message par
                                          un hote (0 <= temps traitement <=
                                          1). 0 = traitement instantané
--hotesTimeoutReemissionMessages        Timeout après lequel les messages
  <Integer>                               doivent etre réexpédiés si aucun
                                          accusé de réception n'est reçu (> 80)
--periodiciteAffichageStats <Float>     Périodicité d'affichage des
                                          statistiques (0 < periodicite <= 1)
```

FIG. 2 – Options disponibles

Ensuite pour spécifier les options, on peut par exemple faire : **java -jar simulation.jar –agentNombreHotes 1000 –duree 5000**.

A.2 Organisation du code source

- Les sources se trouvent dans le dossier **src/main/java**
- Les fichiers de configuration par défaut se trouvent dans le dossier **src/main/resources/configuration**

Le point d'entrée du programme est la classe *Main* qui se trouve dans **src/main/java/be/simulation**.

A.3 Exécution de la simulation

Lancer le programme exécute directement la simulation. Si aucune option n'est spécifiée en argument au programme, les valeurs par défaut sont utilisées. Les résultats sont affichés à l'écran et sauvegardés dans un fichier de log.

A.4 Compilation

La compilation du code requiert l'utilisation de Maven (`http://maven.apache.org/`), un outil de build très simple d'utilisation. En étant dans le dossier du projet (au niveau où se trouve le fichier **pom.xml**), il suffit d'exécuter la commande suivante : **mvn package**. Une fois terminé, le fichier jar exécutable est disponible dans le dossier **target/dist**.

Maven est très simple à installer sur la plupart des distributions Linux (e.g., Ubuntu, ...). Sous Windows, il suffit de le télécharger et d'ajouter le dossier bin au path.