Protocole UGEGreed -- UGP/1.0

Table des matières

Contexte

Dans le cadre du cours de “*Programmation Réseaux*”, il nous a été demandé de créer un protocole de distribution de calcul pour permettre aux chercheurs de tester une conjecture. On le nomme “UGP”.

Introduction

Le protocole **UGEGreed** est un système de calcul distribué qui permet à des applications de partager des tâches en répartissant des calculs sur plusieurs machines. Les applications communiquent* en utilisant le protocole TCP et échangent des fichiers JAR contenant des classes implémentant une interface de vérification de conjectures. Le but de ce protocole est d'aider les chercheurs à tester des conjectures sur un très grand nombre de cas en distribuant leurs calculs sur plusieurs machines.

Terminologies

Les termes suivants sont utilisés dans ce document :

- **Application**: une instance de l'application **UGEGreed** qui peut se connecter à d'autres applications et exécuter des tâches.
- **ROOT**: l'application initiale qui démarre le réseau. Elle accepte des connexions mais ne possède pas de père.
- **Jar:** un fichier contenant les classes Java nécessaires à la vérification des conjectures.
- **Message**: une unité de données envoyée sur le réseau
- **Père:** Machine à laquelle s’est connecté une application
- **Fils**: Machines se connectant à une application
- **Voisins** : L’ensemble des machines avec lesquelles une application peut communiquer, père ou fils
- **Capacité**: Capacité d’une machine sur le réseau de calcul, celle-ci est décidé par l'utilisateur qui implémente le protocole

Spécifications

Connexion

Une application peut être démarrée en mode **ROOT** ou en mode client. Lorsqu'elle est démarrée, elle écoute sur un port donné et attend les connexions d'autres applications. Si l'application est démarrée sans l'adresse d'une autre application, elle est démarrée en mode **ROOT**. Si elle est démarrée avec l'adresse d'une autre application, elle tente de se connecter à cette application.

Réseau

Les applications connectées forment un réseau. Chaque application peut avoir des connexions entrantes et une seule sortante. Lorsqu'une application est connectée à une autre application, elle peut envoyer et recevoir des messages.

Tâches

Pour lancer une tâche, une application doit fournir l'URL du Jar, le nom qualifié de la classe contenue dans le Jar, ainsi que la plage des valeurs à tester. Les tâches à réaliser sont réparties entre les différentes applications du réseau. Chaque application se voit attribuer une plage de valeurs à tester. Lorsqu'une application termine de tester sa plage de valeurs, elle envoie les résultats à l'application qui a fait la demande.
Afin d'assurer une répartition équitable des tâches, chaque machine du réseau possède une table de capacité par tâche en cours et une table de tâches.

Table de tâches

Task_ID	**Receive_From**	**List of (Send_To)**

- **Task_ID :** L'identifiant d'une tâche. Elle est construite de la sorte :

> ***Task_ID*** *= Concaténation (l'adresse IP + Port + Numéro du calcul sur la machine)*

***NB :** À chaque demande, de tâche, une machine crée un identifiant unique pour la tâche en commençant par 1.

- **Receive_From :** L'identifiant de la machine qui demande la tâche
- **List of (Send_To):** Liste des adresse de machine qui accepte la tâche

Table de capacité

Neigh_ID	**Capacity**

- **Neigh_ID :** L'adresse IP de la machine
- **Capacity :** La capacité de cette machine voisine pour une tâche donnée

Messages

Les différents types de messages sont :

Type	Nom	Description
01	**ASK_CAPACITY**	Message de demande de capacité à chaque machine du réseau.
02	**CAPACITY**	Message de réponse à la demande de capacité.
03	**TASK**	Message d'envoie de tâche.
04	**TASK_ACCEPTED**	Message d'acceptation d'une tâche par une machine.
05	**TASK_REFUSED**	Message de refus d'une tâche par une machine.
06	**RESULT**	Message de résultat de la tâche.
07	**LEAVING_NOTIFICATION**	Message de notification de déconnexion au père d'une machine.
08	**NOTIFY_CHILD**	Message de bonne réception de la notification de déconnexion et d'informations au fils de la machine
09	**CANCEL_TASK**	Message d'annulation des tâches aux machines du réseau.
10	**PARTIAL_RESULT**	Message de résultat partiel de la tâche.
11	**NEW_PARENT**	Message de changement de parent aux fils d'une machine en cours de déconnexion
12	**RECONNECT**	Message de reconnexion à une machine en cas de déconnexion
13	**RESUME_TASKS**	Message de continuation de l'exécution des tâches.
14	**ALL_SENT**	

Description du protocol

Connexion au réseau :

Afin de pouvoir distribuer le test de conjecture sur un réseau, une machine rejoint le réseau en utilisant la connexion TCP. Une fois connecté au réseau, il peut envoyer des tests de conjecture aux machines voisines.

Demande de test de conjecture :

Afin de pouvoir partager une tâche au sein d'un réseau, une application à besoin de connaître la capacité de chacun de ses voisins. Pour cela, elle crée une table de capacité pour la tâche qu'elle souhaite réaliser et envoie un message de type "***ASK_CAPACITY*"** à ses machines voisines.

Demande de capacité

Lorsqu'une machine reçoit le message **"*ASK_CAPACITY*"**, elle propage ce message à ses machines voisines, si elle en possède et que le niveau de profondeur dans le paquet est inférieur à **15**. Sinon elle renvoie sa capacité à la machine qui lui a propagé le message. Dans le message propagé, le niveau de profondeur est incrémenté de 1 à chaque propagation. Ci-dessous se trouve le format du message ***"ASK_CAPACITY"*** .

Type (1 octet)	**Task_ID** (10 octets)	**Niveau de profondeur** (1 octet)

- **Type :** 1 (***ASK_CAPACITY***)
- **Task_ID :** L'identifiant de la tâche.
- **Niveau de profondeur:** Afin d'éviter une propagation infinie du paquet de demande de capacité, on inscrit le niveau de profondeur actuelle.

Réponse à la demande de capacité

Pour répondre à un message de demande de capacité (***ASK_CAPACITY***), une machine envoie un message de réponse à la demande de capacité **"*CAPACITY*".** Ce message contient la capacité de cette machine. Lorsque la machine émettrice du message ***ASK_CAPACITY*** reçoit la réponse ***CAPACITY***, elle l'enregistre dans la table de capacité dédiée, puis attend les réponses des autres demandes de capacités. Une fois toutes les réponses reçues, elle additionne les différentes capacités contenues dans la table de capacité et sa propre capacité. Ensuite elle émet un nouveau message ***CAPACITY*** à l'émetteur du paquet ASK_CAPACITY ayant le même Task_ID.. Ce processus se répète jusqu'à la machine à l'origine de demande de capacité. Le format du message **"*CAPACITY*"** est le suivant :

Type (1 octet)	**Task_ID** (10 octets)	**Nb d'octet de la capacité - N** (4 octet)	**Capacité** (N octets)

- **Type :** 2 (***CAPACITY***)

- **Task_ID** : L'identifiant de la tâche.
- **N :** Nombre d'octets de la capacité
- **Capacité:** La capacité de la machine définie à l'implémentation.

Demande de test de conjecture

Une fois que la machine qui a émis la demande de capacité (**ASK_CAPACITY**) reçoit toutes les réponses (**CAPACITY**), elle est en mesure de pouvoir répartir équitablement son test de conjecture et l'envoyer sur le réseau.
Après avoir réparti les charges de travail elle-même en fonction des résultats de la demande de capacité, elle prend sa part de la tâche et envoie le reste aux machines voisines par le message "***TASK***".
Ensuite, elle enregistre dans sa table de tâches, l'identifiant de la tâche, sa propre adresse et la liste des machines à qui elle a distribué la tâche.
Une machine doit, en recevant un message **TASK**, répartir le travail reçu en fonction de sa table de capacité, prendre sa part et envoyer un message **TASK** contenant la part de chaque machine se trouvant dans la table de capacité liée au Task_ID.
Afin d'assurer la retransmission des résultats des tâches, à la réception d'un message de type **TASK,** la machine enregistre dans sa table de tâches l'identifiant de la tâche, son émetteur et la liste des machines à qui elle a distribué la tâche.
Le format message **TASK** est le suivant :

Type (1 octet)	**Task_ID** (10 octets)	**Nb octets URL - N** (4 octets)	**URL du Jar** (N octets)	**Nb octets classe - M** (4 octets)	**Nom de la classe** (M octets)	**FROM** (8 octets)	**TO** (8 octets)

- **Type :** 3 (***TASK***)
- **Task_ID :** Identifiant de la demande de test de conjecture.
- **N :** Nombre d'octets de l'URL en UTF8
- **URL du Jar :** Url du Jar encodé en UTF8
- **M :** Nombre d'octets de la classe à exécuter dans le Jar en UTF8
- **Nom de la classe :** Nom de la classe encodé en UTF8
- **FROM :** Debut de la plage de valeurs à tester en entier 64 bits
- **TO :** Fin de la plage des valeurs à tester en entier 64 bits

Réponse à la demande de test de conjecture

Afin de pouvoir moduler la charge de travail qu'une machine accepte, lorsqu'une machine reçoit un message **TASK,** après propagation aux voisins si nécessaire et enregistrement dans la table de tâche, elle renvoie à l'émetteur un message de type ***TASK_ACCEPTED*** si elle est capable de prendre en charge la plage de valeur qui lui est attribué. Sinon, elle renvoie un message de type ***TASK_REFUSED.*** Lorsque la machine émettrice reçoit ce message, elle prend en charge elle-même la plage de valeur refusée.
Lorsqu'elle aura fini de réaliser sa part de la tâche, elle enverra un message **RESULT** à la machine émettrice de la tâche.

Test de conjecture accepté

Le format du message ***TASK_ACCEPTED*** est le suivant :

Type (1 octet)	**Task_ID** (10 octets)

- **Type :** 4 (***TASK_ACCEPTED***)
- **Task_ID :** Identifiant de la tâche envoyé

Test de conjecture refusé

Le format du message ***TASK_REFUSED*** est le suivant :

Type (1 octet)	**Task_ID** (10 octets)	**FROM** (8 octets)	**TO** (8 octets)

- **Type :** 5 (***TASK_REFUSED***)
- **Task_ID :** Identifiant de la tâche envoyé
- **FROM :** Debut de la plage de valeurs à tester en entier 64 bits
- **TO :** Fin de la plage des valeurs à tester en entier 64 bits

Résultat du test de conjecture

La machine recevant un message **RESULT,** doit regarder la ligne qui correspond à l'identifiant de la tâche reçue dans sa table des tâches.
Ensuite, elle supprime l'adresse de la machine émettrice du message **RESULT** dans la liste des machines pour cette tâche et attend le résultat des autres machines. Une fois tous les résultats des tâches confiées reçues, elle envoie un message **RESULT** à l'émetteur de la tâche contenant la concaténation des différents résultats intermédiaires.
Lorsque la liste des machines d'une des lignes de la table des tâches est vide, la machine peut supprimer cette ligne et la table de capacité liée à la Task_ID.
Si l'émetteur de la tâche est elle-même, cela veut dire que l'ensemble de la conjecture est terminé. Alors, elle écrit dans un fichier l'ensemble des résultats qu'il a stocké.
Le format du message **RESULT** est le suivant :

Type (1 octet)	**Task_ID** (10 octets)	**Nb d'octets du résultat - N** (4 octets)	**Résultat** N octets

- **Type :** 6 (**RESULT**)
- **Task_ID :** Identifiant du résultat de test de conjecture.
- **N :** Nombre d'octets de l'URL en UTF8
- **Résultat :** Concaténation des résultats de la conjecture encodé en UTF8

Déconnexion d'une machine du réseau

Lorsqu'une machine souhaite se déconnecter du réseau, elle doit notifier son parent direct en lui envoyant un message **LEAVING_NOTIFICATION**.

Notification de déconnexion

Le format du message **LEAVING_NOTIFICATION** est le suivant :

Type

(1 octet)

- **Type :** 7 (**LEAVING_NOTIFICATION**)

Confirmation de réception de la notification

Lorsqu'une machine reçoit une notification de déconnexion, elle émet un message **NOTIFY_CHILD** pour dire à la machine émettrice de notifier ses fils.
Ci-après se trouve le format du message **NOTIFY_CHILD.**

Type (1 octet)

- **Type :** 8 (**NOTIFY_CHILD**)

Annulation de la demande de test de conjecture

A la réception du message **NOTIFY_CHILD,** si la machine qui veut se déconnecter est à l'origine d'une demande de conjecture, elle doit envoyer un message **CANCEL_TASK** à chaque machine contenu dans sa table de tâche pour toutes les tâches qu'elle a commencé. Lorsqu'une machine reçoit ce message, elle arrête la tâche liée à l'identifiant se trouvant dans le message et envoie aux machines à qui elle a confié les tâches si elle en a.
Ensuite, elle peut supprimer la ligne contenant le **Task_ID** de la table de tâches et la table de capacité liée à cet id.
Le format du message **CANCEL_TASK** est le suivant :

Type (1 octet)	**Task_ID** (10 octets)

- **Type :** 9 (**CANCEL_TASK**)
- **Task_ID :** Identifiant de la tâche

Résultat partielle du test de conjecture

A la réception du message **NOTIFY_CHILD,** si la machine avait des tâches en cours, elle les arrête, renvoie un message **PARTIAL_RESULT** pour chaque tâche qu'elle fait, au parent et un message **ALL_SENT** pour notifier la fin des transmissions.
Lorsque la machine parent reçoit ce message, elle stocke le résultat. S'il possède le Task_ID dans sa table des tâches, il supprime la machine de la liste des machines pour cette tâche et prend en charge le reste de la tâche.
Sinon, il doit rajouter une ligne dans la table des tâches avec le **Task_ID**, l'émetteur de la tâche et une liste vide qu'il devra remplir avec les adresses des machines qui se connectent avec un message **RECONNECT** ayant un même **Task_ID**.
Le message **PARTIAL_RESULT** contient le résultat partiel des tâches confiées à la machine et son format est le suivant :

Type (1 octet)	**Task_ID** (10 octets)	**Received_From** (4 octets)	**From** (8 octets)	**To** (8 octets)	**Limit** (8 octets)	**Nb d'octets du résultat - N** (4 octets)	**Résultat** N octets

- **Type :** 10 (**PARTIAL_RESULT**)

- **Task_ID :** Identifiant de la requête de demande de tâche.
- **Received_From**: L'adresse de la machine dont provient la tâche
- **From :** Debut de la plage de valeurs de tâche
- **To :** Fin de la partie de la tache faite actuellement
- **Limit:** Fin de la plage de valeurs de la tâche d'origine
- **N :** Nombre d'octets du résultat partiel en UTF8
- **Résultat :** Résultat partiel de la tâche

Le format du message **ALL_SENT** est le suivant :

Type (1 octet)

- **Type :** 14 (**ALL_SENT**)

Changement de parent

Une fois le résultat partiel des tâches envoyé au parent, la machine voulant se déconnecter envoie un message **NEW_PARENT** à ses fils dans lequel se trouve l'adresse de son père direct.
Lorsqu'une machine reçoit ce message, elle met en pause toutes ses tâches, ferme la connexion avec la machine émettrice, met à jour sa table de tâche en remplaçant l'émetteur des taches provenant de la machine émettrice par la nouvelle adresse dans le message.
Finalement elle ouvre une nouvelle connexion à l'adresse et le port contenus dans le message**.**

Type (1 octet)	**Adresse du nouveau parent** (4 octets)	**Port de connexion** (2 octets)

- **Type :** 11 (**NEW_PARENT**)
- **Adresse du nouveau parent:** La nouvelle adresse à laquelle se connecter
- **Port de connexion :** Le port de connexion

Reconnexion au parent

Lorsqu'une machine est déconnectée du réseau alors qu'elle a des tâches en pause, elle doit se reconnecter à la nouvelle adresse en envoyant le message **RECONNECT** . Ce message contient la liste des identifiants des tâches qu'elle a.
Lorsque la machine parent reçoit un message **RECONNECT**, elle regarde pour chacune des identifiants de tâche reçu la ligne associée dans sa table de tâches et ajoute l'adresse de la machine comme "**destinataires des tâches**" pour cette tâche.

Type (1 octet)	**Nb de tâches - N** (4 octets)	**Liste des ID de tâches** (N * 10 octets)

- **Type :** 12 (**RECONNECT**)
- **N :** Nombre des identifiants des tâches.
- **Liste des ID de tâches :** Concaténation des identifiants des tâches sachant qu'un identifiant est 10 octets.

Continuation de l'exécution des tâches

Lorsqu'une machine s’est reconnecté et à envoyer sa liste de tâches, la machine à laquelle elle s’est connectée envoie un message **RESUME_TASKS** pour relancer les tâches mise en pause lors de la déconnexion.
Ci-après se trouve le format du message **RESUME_TASKS.**

Type (1 octet)

- **Type :** 13 (**RESUME_TASKS**)

Références

- Lien du sujet : http://www-igm.univ-mlv.fr/~carayol/coursprogreseauINFO2/tds/projet2023INFO.html
- HTTP RFC : https://www.rfc-editor.org/rfc/rfc1945.txt

Auteurs

Mikdaam BADAROU
Apprenti Ingénieur, ESIPE
Bâtiment Copernic, 5 Bd Descartes, 77420 Champs-sur-Marne

Courriel : bmikdaam@gmail.com

Léo TOSTAIN
ESIPE
Apprenti Ingénieur, ESIPE
Bâtiment Copernic, 5 Bd Descartes, 77420 Champs-sur-Marne

Courriel : leo.tostain@gmail.com