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 applications. 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 applications.

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:** application à laquelle s’est connecté une application
- **Fils**: applications se connectant à une application
- **Voisins** : L’ensemble des applications avec lesquelles une application peut communiquer, père ou fils
- **Capacité**: Capacité d’une application sur le réseau de calcul, celle-ci est décidé par l'utilisateur qui implémente le protocole

Représentation des données

- Les octets signés seront désignés par **[BYTE]** et les suites d'octets par **[BYTES]**.
- Les entiers **[INT]** sur 4 octets signés et les entiers longs **[LONG]** sur 8 octets signés sont tous transmis en Big Endian.
- Les chaînes de caractères **[STRING]** sont encodées en ***UTF_8*** et précédées de la taille de leur représentation en octets sur un INT.

  [STRING] = taille **[INT]** chaîne encodée en UTF-8 **[BYTES]**

- Une adresse IP **[IPADDRESS]** est représentée par un octet valant 4 ou 6 selon que l'adresse est au format IPv4 ou IPv6 suivi des 4 octets ou 16 octets de l'adresse selon qu'elle est IPv4 ou IPv6.
- Une adresse de socket **[SOCKETADDRESS]** contenant une adresse IP et un numéro de port est représentée un **IPADDRESS** suivi d'un INT valant entre 0 et 65 535.

  [SOCKETADDRESS] = adresse IP [**IPADDRESS**] + numéro de port entre 0 et 65 535 [**INT**]

- Les urls http **[HTTP-URL]** référencent des **[STRING]** qui suivent la RFC 4248 pour les URLs pour le schéma HTTP.
- Les identifiants **[ID]** seront simplement des **[SOCKETADDRESS]**
- Les identifiants de tâches **[TASK_ID]** sont des **[SOCKETADDRESS]** précédés de l'identifiant de la tâche pour l'application

  [TASK_ID] = adresse de socket **[SOCKETADDRESS]** + identifiant **[LONG]**

- Les listes d'identifiants de tâche **[LIST_ID]** seront envoyées au format suivant un **INT** donnant le nombre d'identifiants dans la liste suivi des identifiants de la liste au format **[TASK_ID]**

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 application du réseau possède une table de capacité par tâche en cours et une table de tâches.

Table de tâches

TASK_ID	**Soure [ID]**	**Destinations [LIST_ID]**

- **Source:** L'identifiant de l'application qui demande la tâche
- **Destinations:** Liste des adresse de application qui accepte la tâche

Table de capacité [par tâche]

Neighbor [ID]	**Capacity**

- **Neighbor:** L'adresse IP de l'application
- **Capacity :** La capacité de cette application voisine pour une tâche donnée

Messages

Les différents types de messages sont :

Type	Nom
01	**CAPACITY_REQUEST**
02	**CAPACITY**
03	**TASK**
04	**TASK_ACCEPTED**
05	**TASK_REFUSED**
06	**RESULT**
07	**LEAVING_NOTIFICATION**
08	**NOTIFY_CHILD**
09	**CHILD_NOTIFIED**
10	**CANCEL_TASK**
11	**PARTIAL_RESULTS**
12	**NEW_PARENT**
13	**NEW_PARENT_OK**
14	**RECONNECT**
15	**RECONNECT_OK**

Description du protocol

Connexion au réseau :

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

Soumission d'une tâche au réseau :

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 "***CAPACITY_REQUEST*"** à ses applications voisines.

Demande de capacité pour une tâche

Lorsqu'une application reçoit le message **"*CAPACITY_REQUEST*"**, elle propage ce message à ses applications voisines, si elle en possède. Sinon elle renvoie sa capacité à l'application 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 ***"CAPACITY_REQUEST"*** .

Type(1) **[BYTE]**	Id **[TASK_ID]**

- **Type :** 1 (***CAPACITY_REQUEST***)
- **Id:** L'identifiant unique de la tâche de la tâche à lancer

Réponse à la demande de capacité

Pour répondre à un message de demande de capacité (***CAPACITY_REQUEST***), une application envoie un message de réponse à la demande de capacité **"*CAPACITY*".** Ce message contient la capacité de cette application pour la tâche. Lorsque l'application émettrice du message ***CAPACITY_REQUEST*** 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 **CAPACITY_REQUEST** ayant le même **TASK_ID**. Ce processus se répète jusqu'à l'application à l'origine de demande de capacité. Le format du message **"*CAPACITY*"** est le suivant :

Type(2) **[BYTE]**	Id **[TASK_ID]**	**Nb d'octet de la capacité - N** (4 octet)	**Capacité** (N octets)

- **Type :** 2 (***CAPACITY***)
- **Id**: L'identifiant de la tâche.
- **N :** Nombre d'octets de la capacité

- **Capacité:** La capacité de l'application définie à l'implémentation.

Attribution des parts de tâche sur le réseau

Une fois que l'application qui a émis la demande de capacité (**CAPACITY_REQUEST**) 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 applications 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 applications à qui elle a distribué la tâche.
Une application 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 application 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,** l'application enregistre dans sa table de tâches l'identifiant de la tâche, son émetteur et la liste des applications à qui elle a distribué la tâche.

*Le format message **TASK** est le suivant :*

Type(3) **[BYTE]**	Id **[TASK_ID]**	URL **[HTTP-URL]**	Nom de la classe qualifié **[STRING]**	From **[LONG]**	To **[LONG]**

- **Type :** 3 (***TASK***)
- **Id:** Identifiant de la demande de test de conjecture.
- **Url:** Url du Jar encodé 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 l'attribution

Afin de pouvoir moduler la charge de travail qu'une application accepte, lorsqu'une application 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 l'application 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 l'application émettrice de la tâche.

Tâche accepté

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

Type(4) **[BYTE]**	Id **[TASK_ID]**

- **Type :** 4 (***TASK_ACCEPTED***)

- **Id:** Identifiant de la tâche envoyé

Tâche refusé

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

Type(5) **[BYTE]**	Id **[TASK_ID]**	From **[LONG]**	To **[LONG]**

- **Type :** 5 (***TASK_REFUSED***)
- **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 d'une tâche acceptée

Lorsqu'une application reçoit le résultat d'une tâche [**RESULT**], elle l'envoie à l'émetteur de la tâche qui se trouve dans sa table des tâches. Si l'émetteur de la tâche est elle-même, cela signifie que l'ensemble de la conjecture est terminé. Dans ce cas, elle écrit le résultat dans un fichier dédié aux résultats.
Le format du message **RESULT** est le suivant :

Type(6) **[BYTE]**	Id **[TASK_ID]**	Résultat **[STRING]**

- **Type :** 6 (**RESULT**)
- **Id :** Identifiant de la tâche
- **Résultat :** Concaténation des résultats de la conjecture encodé en UTF8

Déconnexion d'une application du réseau

Lorsqu'une application souhaite se déconnecter du réseau, elle:

- se mets en état de non disponibilité[c'est-à-dire elle n'accepte plus aucune tâche]
- arrête et supprime les tâches lancées par elle-même et les parts de la tâche données aux voisins en leur envoyant un message **CANCEL_TASK**
- arrête et envoie les résultats partiels des tâches acceptées à leur émetteur par le message **PARTIAL_RESULTS**
- notifie son parent direct en lui envoyant un message **LEAVING_NOTIFICATION**

Annulation des tâches distribuées au réseau

Lorsqu'une application reçoit ce message, elle arrête la tâche liée à l'identifiant se trouvant dans le message et envoie aux applications 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(10) **[BYTE]**	Id **[TASK_ID]**

- **Type :** 10 (**CANCEL_TASK**)
- **Id:** Identifiant de la tâche

Résultat partielle des tâches acceptées

Lorsque l'application émettrice reçoit ce message, elle le transmet à l'émetteur de la tâche, supprime l'application de la liste des applications destinations pour cette tâche et prend en charge le reste de la tâche.
Le message **PARTIAL_RESULTS** contient le résultat partiel des tâches confiées à l'application et son format est le suivant :

Type(11) **[BYTE]**	Number **[INT]**	Id **[TASK_ID]**	From **[LONG]**	To **[LONG]**	Stopped_at **[LONG]**	Résultat **[STRING]**

- **Type :** 11 (**PARTIAL_RESULTS**)
- **Number:** Nombre de résultats partiels dans le paquet
- **Id:** Identifiant de la requête de demande de 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
- **Résultat :** Résultat partiel de la tâche

Notification de déconnexion

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

Type(7) **[BYTE]**

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

Confirmation de réception de la notification

Lorsqu'une application reçoit une notification de déconnexion[**LEAVING_NOTIFICATION**], elle répond avec un message **NOTIFY_CHILD** pour dire à l'application émettrice de notifier ses fils de la déconnexion.
Ci-après se trouve le format du message **NOTIFY_CHILD.**

Type(8) **[BYTE]**

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

Changement de parent

Quand l'application voulant se déconnecter reçoit le message **NOTIFY_CHILD,** elle envoie un message **NEW_PARENT** à toutes ses applications filles si elle en possède, sinon elle répond par **CHILD_NOTIFIED**.
A la réception de **NEW_PARENT** par une application fille, elle ouvre une nouvelle connexion à l'adresse du nouveau parent. Ensuite, envoie un message **RECONNECT** au parent.
Lorsque la reconnexion avec le parent est bien établie [connue grâce à la réception du message **RECONNECT_OK**], l'application répond à l'application émettrice un message **NEW_PARENT_OK**.

Le format du message **NEW_PARENT** est :

Type (12) **[BYTE]**	Parent_addr

	[SOCKETADDRESS]

- **Type :** 12 (**NEW_PARENT**)
- **Parent_addr :** Adresse du nouveau parent

Le format de **CHILD_NOTIFIED** est :

Type(9) **[BYTE]**

Le format du message **NEW_PARENT_OK** est :

Type(13) **[BYTE]**

Reconnexion au parent

Lorsque l'application parent reçoit un message **RECONNECT**, elle regarde pour chacune des identifiants de tâche reçu la ligne associée dans sa table de tâches et remplace l'adresse de l'application voulant se déconnecter par l'adresse se trouvent dans le message. Enfin, elle répond avec le message **RECONNECT_OK.**
Le format du message **RECONNECT** est :

Type(14) **[BYTE]**	child_addr **[SOCKETADDRESS]**

- **Type :** 14 (**RECONNECT**)
- **child_addr :** Adresse de l'application qui fait la reconnexion

Le format du message **RECONNECT_OK** est :

Type(15) **[BYTE]**

Reconnexion réussie

Lorsque le message **RECONNECT_OK** est reçu par l'application qui fait une reconnexion, elle informe son ancien parent en envoyant un message **NEW_PARENT_OK.**

Lorsque l'ancien parent reçoit ce dernier, il attend la réponse des autres applications filles, ensuite informe son père en envoyant un message **CHILD_NOTIFIED.**

Fermeture de connexion

Lorsqu'une application reçoit le message **CHILD NOTIFIED** ou **NEW_PARENT_OK**, la connexion est automatiquement fermée entre l'application émettrice et elle-même.

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