1 Introduction

1.1 Objectifs du protocolle

Le protocole CALC permet la communication entre un client et un serveur de calcul au travers d'un réseau TCP/IP. Le client doit pouvoir soumettre des opération artihmétiques au serveur et recevoir la réponse. Chaque serveur doit au moins implémenter l'addition et la soustraction de deux opérandes. Il peut offrir des opérations supplémentaires qui seront communiquées au client lors de la connexion.

Fonctionnement

Le protocole CALC utilise le protocole de transport TCP sur le port 3193. Après avoir accepté une demande de connexion le serveur envoie la liste des opérations supportées au client. Le client peut envoyer autant de requêtes de calcul qu'il souhaite.

Chaque demande de calcul est faite grâce au message `COMPUTE` suivit de l'opération demandée et des opérandes requises. Si le serveur reçoit un message `COMPUTE` correct il calcule le résultat et le renvoie au client grâce à un message `RESULT`. En cas d'erreur il revoie un message `ERROR`.

Si le client n'a plus de requêtes à faire il envoie un message `QUIT` et ferme la connection TCP. Le protocole CALC est sans état.

2 Syntaxe

Le protocole CALC définit 6 messages : `NOTIFY`, `HELP`, `COMPUTE`, `RESULT`, `ERROR` et `QUIT`. Le protocole est un protocole "ligne par ligne" qui utilise la séquence *CRLF* pour indiquer une fin de ligne. Chaque élément du message est séparé par un espace blanc.

2.1 `NOTIFY`

Au moment de la connexion du client au serveur, ce dernier envoie un message `NOTIFY` au client avec la liste des opérations supportées. S'ensuit une énumération ligne par ligne des opérations supportées par le serveur. Chaque ligne contient le nom de l'opération ainsi que le nombre d'opérandes à envoyer. Le message est terminé par une ligne composée du texte `END NOTIFY`.

Exemple

NOTIFY *CRLF*
ADD 2 *CRLF*
SUB 2 *CRLF*
MUL 2 *CRLF*
DIV 2 *CRLF*
POW 2 *CRLF*
END NOTIFY *CRLF*

2.2 HELP

Le client peut a tout moment demander de l'aide sur une opération. La syntaxe est la suivante : `HELP <OPERATION> <NB D'OPERANDES>`. Le serveur va renvoyer un message similaire en ajoutant la description de l'opération à la fin du message.

Exemple

Client : HELP ADD 2 *CRLF*
Serveur : HELP ADD 2 Additionne les deux nombres ensemble *CRLF*

2.3 COMPUTE

Le client peut envoyer une demande de calcul au serveur grâce au message `COMPUTE`. Le message est composé de l'opération voulue ainsi que des opérandes requises.

Exemple

COMPUTE DIV 10 2 *CRLF*

2.4 RESULT

Le serveur renvoie ce message après avoir reçu une demande `COMPUTE` de la part du client. Le message contient le résultat du calcul demandé.

Exemple

RESULT 5 *CRLF*

2.5 ERROR

Dans le cas où le client envoie un message qui ne peut pas être intérprété par le serveur celui-ci envoie un message d'erreur. Ce dernier est définit par la syntaxe suivante : `ERROR <CODE> <DESCRIPTION DE L'ERREUR>`.

Exemple

ERROR 404 COMMAND NOT FOUND *CRLF*

2.6 `QUIT`

Ce message est envoyé par le client pour indiquer au serveur qu'il n'a plus besoin de ses services et que la session est terminée.

3 Eléments spécifiques

3.1 Opérations supportées

Un serveur doit au moins implémenter les opérations `ADD 2` et `SUB 2`, représentant respectivement l'addition et la soustraction de deux éléments.

3.2 Gestion des erreurs

La gestion des erreurs s'inspire des status HTTP. En cas de requête invalide du client, le serveur doit pouvoir répondre avec le code d'erreur correspondant. Les codes possibles sont les suivants :

- `400 BAD REQUEST` : Le client a envoyé une requête comportant une erreur de syntaxe
- `404 NOT FOUND` : Le client a fait une demande d'opération ou de commande non supportée par le serveur
- `500 INTERNAL SERVER ERROR` : La requête du client a mené à une erreur du côté serveur. Ceci peut être du à des opérations mathématiques interdites comme une division par zéro. La requête ne doit pas être retentée telle quelle.

3.3 Extensibilité

Si un serveur veut implémenter des opérations supplémentaires il doit les envoyer au client lors de la connexion à l'aide du message `NOTIFY` en respectant la syntaxe défini ci-dessus.

4 Exemples

Exemple 1

S: NOTIFY *CRLF*
S: ADD 2 *CRLF*
S: SUB 2 *CRLF*
S: MUL 2 *CRLF*
S: POW 2 *CRLF*
S: END NOTIFY *CRLF*
C: COMPUTE MUL 2 3 *CRLF*
S: RESULT 6 *CRLF*
C: QUIT

Exemple 2

S: NOTIFY *CRLF*
S: ADD 2 *CRLF*
S: SUB 2 *CRLF*
S: POW 2 *CRLF*
S: END NOTIFY *CRLF*
C: HELP POW 2 *CRLF*
S: HELP POW 2 Met la première operande a la puissance de la deuxieme *CRLF*
C: QUIT

Exemple 3

S: NOTIFY *CRLF*
S: ADD 2 *CRLF*
S: SUB 2 *CRLF*
S: DIV 2 *CRLF*
S: END NOTIFY *CRLF*

C: COMPUTE ADD 5 *CRLF*
S: ERROR 400 BAD REQUEST *CRLF*
C: ADD 2 8 *CRLF*
S: ERROR 404 NOT FOUND *CRLF*
C: COMPUTE POW 2 8 *CRLF*
S: ERROR 404 NOT FOUND *CRLF*
C: COMPUTE DIV 10 0 *CRLF*
S: ERROR 500 INTERNAL SERVER ERROR *CRLF*
C: QUIT *CRLF*