Cherokee ETNA

Ce document a pour but de démontrer la démarche que notre groupe a eu, afin de mettre en place au mieux le projet Cherokee

Les composants

Nous avons tout d'abord définie les paramètres des différents composants :

- Le port d'écoute (12345)
- La taille du buffer (4096)
- La taille du thread pool de notre programme (20)
- Un mutex de thread (voir partie optimisation et multi-connexion)
- Un signal de condition de thread (voir partie optimisation et multi-connexion)

Nous avons aussi décidé de séparer notre programme en 4 fonctions distinctes :

- La fonction main(), qui est la fonction qui va s'exécuter lorsque l'on lance le programme
- La fonction check(), qui est la fonction qui va nous permettre de savoir si une erreur se produit sur notre socket
- La fonction thread_function(), qui est la fonction qui va nous permettre de gérer et optimiser la gestion de nos threads (voir partie optimisation et multi-connexion)
- La fonction handle_connection(), qui va gérer la réponse et le parsing de celle ci, ainsi que les différentes méthodes tel que GET, POST, DELETE etc..

Gérer les multi-connexions et l'optimisation

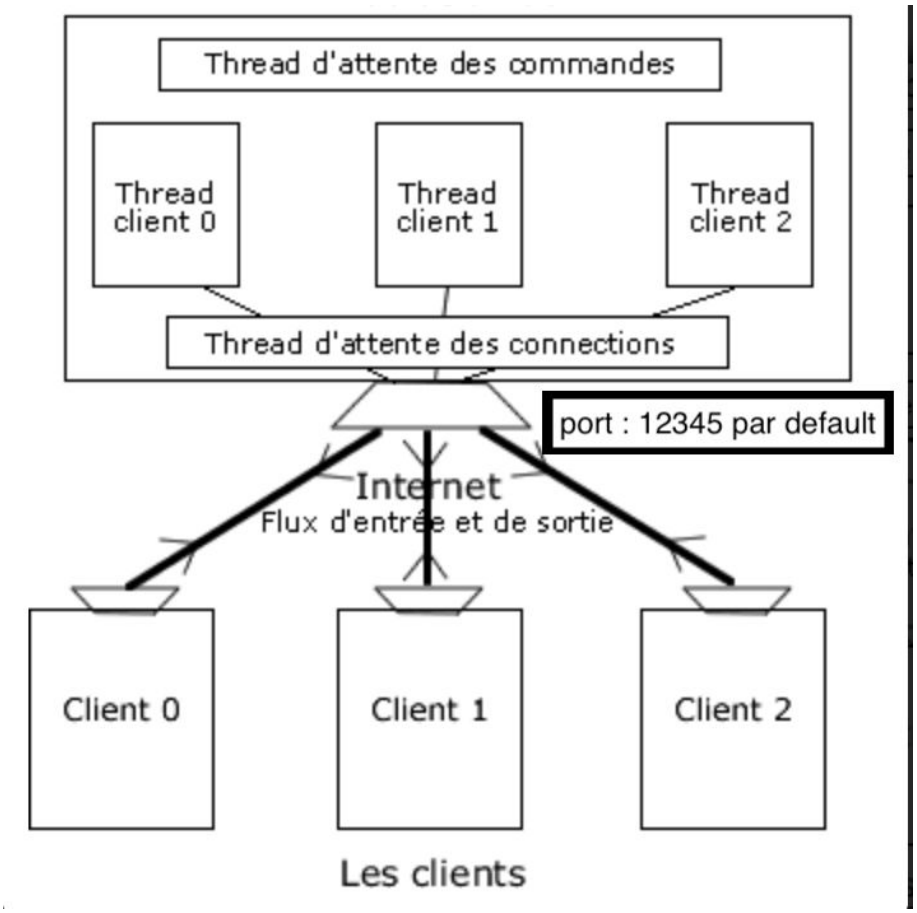
Nous avons voulu gérer au maximum les multi-connexions ainsi que l'optimisation de la mémoire avec une gestion des threads.
Pour ce faire nous utilisons différents composants tels que les conditions de signaux, les listes chaînées ainsi que les mutex.

Comment fonctionne tout cela?

Lorsqu'un nouveau client se connecte un système de synchronisation se lance aussi appelé mutex afin de ne pas endommager la mémoire, par la suite nous allons rajouter

ce même client dans une liste chainé grâce aux fonction (enqueue / dequeue), la liste chainé aussi appélé noeud est composé lui même d'un pointeur qui point un autre noeud ainsi qu'un client. Les fonctions enqueue et dequeue permettent d'extraire le socket_client à la tête ou de rajouter un nouveau noeuds. Evidemment le derniers noeuds de cette liste est un pointeur nul. Pour optimiser le code, un système de signaux a été mis en place afin de mettre en attente des threads. Pour ce faire lorsqu'il y a déjà un client en traitement, nous avons mis en place le cond wait qui attends la réponse du signal afin d'extraire un le client de l'entête initialement ajouté.

Diagramme explicatif



La structure http et les methodes

Afin de gérer au mieux les différentes méthodes, nous avons mis en place un structure de classe http. Celle ci comprend une fonction appelé fill_request() qui va nous permettre de retrouver la méthode ainsi que l'url. Ensuite on retrouve une fonction par méthode, get, post, put etc.. Chaques fonctions méthodes possèdent ses propres caractéristiques. Par exemple la fonction getFonction va tout d'abord recevoir le client, ensuite il va ouvrir le fichier présent dans l'url en le parsant, une fois ouvert et qu'aucune erreur n'a été détecté, il lit le contenu du fichier et l'envoie au client puis ferme la connexion.