Message Queues – Spécification et Design

Les méthodes sont abstract. Pour utiliser les méthodes des classes abstraites, utiliser une sous classe concrète de celles ci.

Class QueueBroker : Permet de créer des MessageQueue utilisable. connect et accept peuvent s'effectuer indépendamment de l'ordre dans lequel ils sont appelés.

QueueBroker : Constructeur de la classe, permet de créer une instance de QueueBroker.
Paramètres :
- name (String) : le nom du QueueBroker

Return : QueueBroker

accept : Le QueueBroker se met en attente d'une connexion sur le port précisé. Méthode bloquante tant que le rendez-vous n'est pas accepté. Le messageQueue est alors crée.
Parameters:
port (int) – le port sur lequel le QueueBroker se met en attente de connexion
Returns :
le MessageQueue

connect : Connexion à un autre MessageQueue dont le nom est donné sur le même port précisé. Méthode bloquante tant que le rendez-vous n'est pas accepté. Le messageQueue est alors crée.
Parameters:
port (int) – le port sur lequel le QueueBroker veut se connecter
name (string) – le nom du QueueBroker auquel se connecter
Returns :
le MessageQueue

Class MessageQueue : Permet d'échanger des messages donc échange de flux de paquets.

send : Envoie la partie de bytes entre offset et offset+length. Bloquant en attendant que tous les bytes soient envoyés.
Parameters:
bytes (byte[]) - buffer contenant les octets à écrire
offset (int) – l'offset de debut
length (int) - le nombre d'octets à écrire

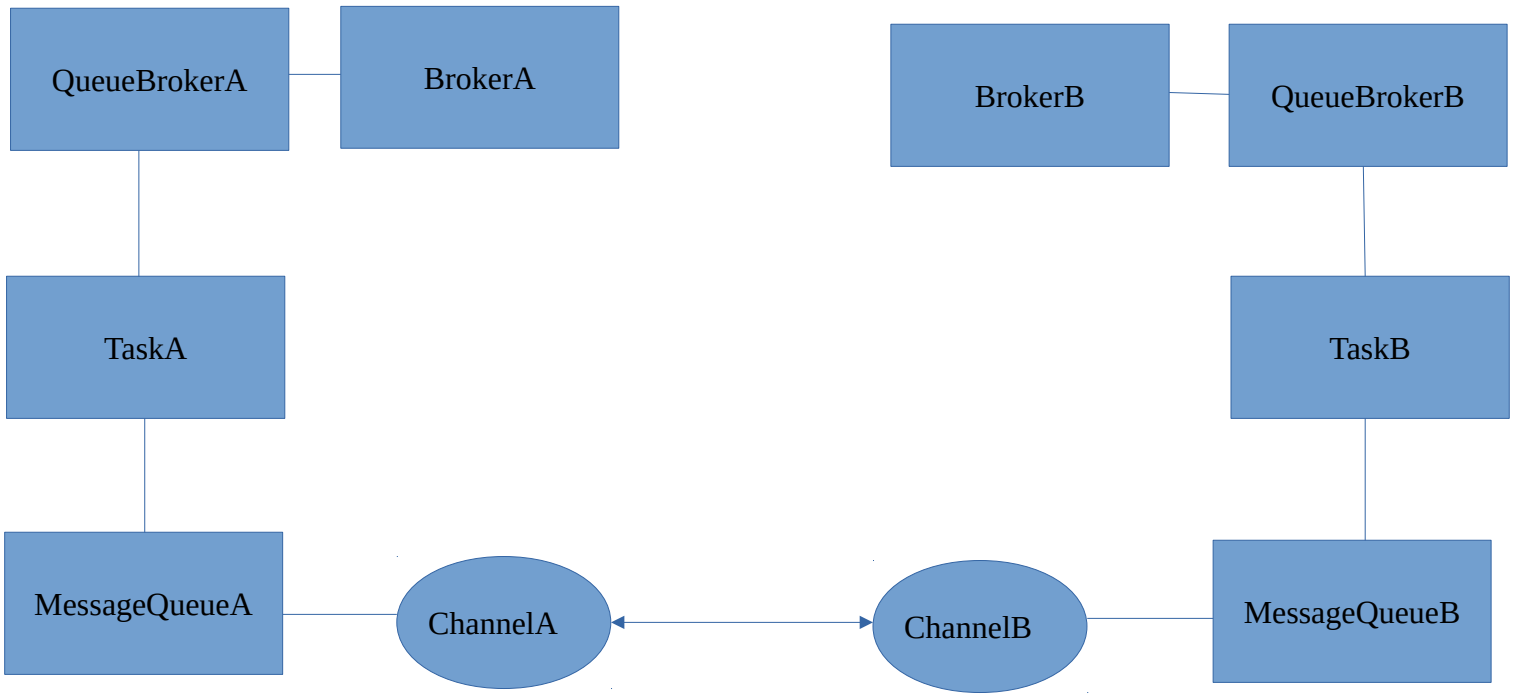
receive : Renvoie le tableau d'octets correspondant au message que l'autre à envoyé. Bloquant tant que tout n'est pas lu.
Returns : Le tableau d'octets contenant le message.

close : Ferme la connexion entre deux MessageQueue donc déconnecte les deux MessageQueue impliqué dans la connexion.

closed : Permet de savoir si le MessageQueue est déconnecté. Si c'est le cas return true.
Returns :
Un boolean indiquant si le MessageQueue est déconnecté.

Synchronisation des Threads : messageQueue est partagé entre plusieurs threads il faut donc une synchronisation. En effet, chaque paquet a un sens indépendamment des autres paquets. Il n'est pas nécessaire de savoir d'où les paquets viennent. Ils se comportent comme une suite de messages cohérent.
Ils faut synchroniser les méthodes send et receive.

Task crée un QueueBroker qui crée un Broker qui crée un Channel. Le QueueBroker crée un MessageQueue et lui passe le Channel créé.



Forme du message : entête contenant la longueur sur 4 octets puis le contenu correspondant au message.
Send bloque la ressource, écrit la taille du message, écrit le messages, délivre la ressource
Receive bloque la ressource, lit les 4 premiers octets afin de connaître la taille du message, lit le message et libère la ressource.