# TP2: FAQ par [François Pierre Doray](https://moodle.polymtl.ca/user/view.php?id=29204&course=612)

## Pourquoi est-ce que j'obtiens l'erreur "RemoteException occured in server thread; nested exception is: java.rmi.AccessException: Registry.Registry.rebind disallowed; origin /132.207.xx.xx is non-local host"?

Comme on peut le voir dans la documentation de la méthode Registry.bind [http://docs.oracle.com/javase/7/docs/api/java/rmi/registry/Registry.html#bind(java.lang.String,%20java.rmi.Remote)](http://docs.oracle.com/javase/7/docs/api/java/rmi/registry/Registry.html#bind%28java.lang.String,%20java.rmi.Remote%29) , il n'est pas possible d'appeler la méthode bind() sur un rmiregistry distant\*. Il faut donc:

* Lancer un rmiregistry local pour chaque machine servant de calculateur.
* Enregistrer chaque interface de calculateur auprès du rmiregistry *local*.
* À partir du répartiteur, se connecter aux différents rmiregistry exécutés sur les machines des calculateurs pour obtenir les interfaces des calculateurs.

\* Il existe des work-arounds, mais je ne suggère pas de les utiliser pour ce TP. Si vous les utilisez, assurez-vous d'expliquer clairement les étapes nécessaires pour faire fonctionner votre TP dans le fichier Readme demandé.

## Pourquoi est-ce que j'obtiens l'erreur "Exception creating connection to: 132.207.12.174; nested exception is: java.net.NoRouteToHostException: Aucun chemin d'accès pour atteindre l'hôte cible"?

Dans les laboratoires de Polytechnique, seuls les ports entre 5000 et 5050 sont ouverts. L'annexe de l'énoncé détaille les précautions que vous devez prendre pour que toutes vos communications soient faites sur ces ports.

En résumé:

* Démarrez le rmiregistry en spécifiant un port autorisé.
  + Exemple: /opt/java/jdk8.x86\_64/bin/rmiregistry **5001**
* Accédez au rmiregistry en spécifiant ce même port. (Vous devrez probablement faire cette étape à la fois dans le code du répartiteur et des calculateurs).
  + Exemple: Registry registry = LocateRegistry.getRegistry(**5001**);
  + Documentation: [http://docs.oracle.com/javase/7/docs/api/java/rmi/registry/LocateRegistry.html#getRegistry(int)](http://docs.oracle.com/javase/7/docs/api/java/rmi/registry/LocateRegistry.html#getRegistry%28int%29)
* Exportez votre objet en spécifiant un port autorisé.
  + Exemple:
  + ServerInterface stub = (ServerInterface)UnicastRemoteObject.exportObject(this, **5002**);
  + Documentation: [http://docs.oracle.com/javase/7/docs/api/java/rmi/server/UnicastRemoteObject.html#exportObject(java.rmi.Remote,%20int)](http://docs.oracle.com/javase/7/docs/api/java/rmi/server/UnicastRemoteObject.html#exportObject%28java.rmi.Remote,%20int%29)

## Quel est le taux de refus des calculateurs lorsque la formule T= (ui−qi)/9qi∗100% ne donne pas une valeur entre 0 et 100%?

À partir du moment où u\_i est 10 fois plus grand que q\_i, la formule donne une valeur plus grande que 100. On considère alors que la tâche est refusée 100% du temps. Aussi, lorsque u\_i est plus petit que q\_i, la tâche est toujours acceptée.

Entre ces bornes, on utilise la formule pour calculer la probabilité de refus. Vous pouvez utiliser la librairie standard de Java pour générer un nombre aléatoire entre 0 et 100. Si ce nombre est plus petit que la valeur calculée pour T, vous refusez la tâche. Sinon, vous l'acceptez.

## En mode non-sécurisé, devons-nous considérer le cas où plus de 50% des calculateurs seraient malicieux?

Non, vous pouvez en tout temps considérer que plus de 50% des serveurs sont honnêtes. Aussi, en cas de panne, vous pouvez considérer qu'il reste toujours plus de 50% de serveurs honnêtes.

## Pouvons-nous avoir plus de détails sur le mode non-sécurisé?

En mode non-sécurisé, chaque bloc d'opérations doit être soumis à un vote démocratique impliquant au moins 2 calculateurs.

Ainsi, vous pouvez dans un premier temps envoyer chaque bloc à 2 calculateurs. Si les 2 calculateurs retournent la même réponse, vous retenez cette réponse (vous avez obtenu un appui d'au moins 50%). Sinon, vous devez transmettre le bloc à d'autres calculateurs jusqu'à ce qu'au moins 50% des calculateurs consultés aient produit la même réponse. Cette condition sera toujours respectée si plus de 50% des calculateurs sont honnêtes.

## Comment gérer les défaillances en mode non-sécurisé?

Vous devez toujours respecter le vote démocratique impliquant au moins 2 calculateurs et pour lequel au moins 50% des calculateurs consultés ont retourné la même réponse. Si un calculateur tombe en panne alors qu'il traitait un bloc d'opérations, vous devez retransmettre ce bloc à un autre calculateur. Un calculateur ne peut pas voter plusieurs fois pour un même bloc.

## Peut-on envoyer les opérations une par une aux calculateurs?

Tel que nous l'avons vu au TP1, il y a un surcoût associé aux appels RMI. Ainsi, si vous envoyez les opérations une par une, vous devrez payer ce surcoût pour chaque opération. Vous risquez d'avoir de meilleures performances en envoyant des blocs d'opérations.

Dans votre rapport, discutez des avantages et des inconvénients d'utiliser des gros blocs d'opérations. Dans votre implémentation, choisissez une taille initiale raisonnable pour les blocs d'opérations (elle peut être proportionnelle à la taille du fichier à traiter). Vous devrez sans doute ajuster dynamiquement cette taille en fonction des refus renvoyés par les calculateurs lorsqu'ils manquent de ressources.

## Rappel

Vous devez remettre un fichier README expliquant précisément comment exécuter chacun des tests de performance demandés dans ce TP dans le laboratoire L-4714.

## Pour les tests de performance en mode non-sécurisé, est-ce que je devrais obtenir des résultats différents pour les 3 cas à tester?

Oui. Pour ces tests de performance, il est spécifié que vous devez instancier la même configuration que pour les tests en mode sécurisé, c'est à dire que vous devez respecter la relation q\_1 = 2\*q\_2 = 4\* q\_3 entre les 3 calculateurs. Le serveur malicieux aura donc une valeur q\_i différente dans chacun des 3 cas que l'on demande de tester. Cela devrait avoir un impact sur la performance globale du calcul.

## Devons implémenter un répartiteur capable de gérer N calculateurs, ou un répartiteur sachant qu'il dispose d'exactement 3 calculateurs?

Le répartiteur doit être capable de gérer n'importe quel nombre de calculateurs.

## Doit-on gérer les défaillances de serveurs honnêtes en mode non-sécurisé?

Oui, vous devez gérer tous les cas où il reste *plus de 50%* de calculateurs honnêtes.

Par exemple, si on démarre le système avec 3 calculateurs honnêtes et 1 calculateur malicieux, vous devez gérer la défaillance d'un des calculateurs honnêtes (il resterait alors 66% de calculateurs honnêtes). Cependant, si on a initialement 2 calculateurs honnêtes et 1 calculateur malicieux (comme dans les tests de performance demandés), vous n'avez pas à gérer la panne d'un calculateur honnête (il resterait seulement 50% de calculateurs honnêtes).

## En mode non-sécurisé, peut-on envoyer chaque requête à tous les calculateurs et prendre la réponse ayant obtenu le plus de votes?

Veuillez consulter la réponse à la question «Pouvons-nous avoir plus de détails sur le mode non-sécurisé?»ci-dessus. Votre implémentation devrait être pensée pour offrir une meilleure performance lorsqu'il y a plus de calculateurs. Si vous envoyez chaque requête à 100% des calculateurs, vous n'aurez pas une meilleure performance en ajoutant des calculateurs... C'est pourquoi on demande plutôt d'envoyer chaque bloc d'opérations à 2 calculateurs dans un premier temps. Si les 2 retournent la même réponse, vous la retenez. Sinon, vous transmettez le bloc d'opérations à un autre calculateur jusqu'à ce qu'au moins 50% des calculateurs aient retourné la même réponse.

Vous n'avez pas à gérer le cas où il y a moins de 50% de calculateurs honnêtes. Ainsi, vous aurez toujours une réponse ayant 50% d'appui après avoir consulté tous les calculateurs (pire cas).