SYSTÈME MOBILE
SYM

PROTOCOLE APPLICATIF

Domingues Pedrosa João Miguel
Haas Loïc

6 novembre 2015

Table des matières

1 Comportement de l'application

L'application que nous avons crée, sert à tester les différentes manipulation du laboratoire. Il s'agit donc d'envoyer des données de différents format (String, JSON, compressée) de manière asynchrone. Pour cela, nous avons 3 bouton qui envoie chacune des informations selon différents format.

- Envoyer Basic : envoie juste une chaîne de caractère sans traitement quelconque.
  Message : Super Salut

- Envoyer JSON : envoie des données sous format JSON.
  Message : "MyValue" :"doge", "MyValue2" :"Super salut"

- Envoyer ZipBase64 : envoie des données compresser en Base64.
  Message : Super Salut text

Lorsque le message est envoyer, on attend la réponse du serveur avec EventListener afin que l'on puisse continuer à utiliser l'application sans avoir à attendre le retour. Une fois la réponse réceptionner, on affiche le contenu dans la fenêtre d'application. Lorsque la réponse est envoyer sous forme compresser, on applique une décompression afin d'avoir un message compréhensible afficher. Dans les autres cas, aucun traitement en particulier est appliquer si ce n'est que l'affichage.

2 Authentification

Il est tout à fait possible de transmettre de manière asynchrone même avec une authentification requise. La restriction serait que pour tout interaction avec l'utilisateur, il faudrait d'abord envoyer une notification si on ne se trouve pas directement dans l'application. Premièrement, pour indiquer à l'utilisateur qu'une validation de sa part est nécessaire afin de continuer le bon fonctionnement de l'application. Deuxièmement, il est interdit sur android de lancer une activité sans que l'utilisateur n'est rien demander. Pour la transmission en différé, si tout a été confirmer avant, il enverra une fois que la connexion au serveur sera possible car dans les données à envoyer, on devrait au préalable avoir les différents champs pour l'authentification.

3 Threads concurrents

Il y des problème de synchronisation lors de la modification de l'interface graphique. Il peut y avoir des accès concurrent quand aux données entre l'envoie et la réception. Par exemple, une variable utilisé par les deux threads qu'il faudrait protéger afin d'éviter des problème d'incohérences quand elle est manipuler. Un autre problème serait que lorsqu'on envoie un message, on attendrai une réponse. Il se peut quand même temps qu'on envoie on reçoit quelque chose et que l'on traite la réception pour le message juste envoyer alors que cela n'a rien à voir, donc problème de synchronisation entre les deux threads.

4 Écriture différée

Voici nos proposions d'implémentation quand on a plusieurs écriture en attente et qu'il devient tout d'un coup possible de les envoyer simultanément.

La première méthode serait de passer par un queue où chaque écriture attend son tour. Pour chaque transmission, on ouvre la connections, on envoie puis on la referme et on passe au prochaine envoie, s'il y en a un autre.

Dans le deuxième cas, on utilise la même queue mais on ouvre d'abord la connexion, on fait toute les transmissions nécessaire puis on referme, au lieu d'ouvrir une nouvelle connexion pour chaque envoie. L'avantage, c'est que l'on n'a pas à chaque fois à ouvrir une connexion, on envoie tout dans une seule

ce qui est plus rapide et moins contraignant. Quand au réponse du serveur, on peut gérer ça avec un EventListener se chargeant de traiter la donnée quand on la reçoit et de les placer dans un buffer pour un traitement ultérieur.

L'avantage de la première solution, c'est qu'a chaque écriture correspond une transmission, il est donc plus facile lorsque l'on reçoit une réponse de la gérer car on sait à quel connexion correspond le message. Dans le deuxième, l'avantage est que l'on est plus efficace, on ouvre un seule connexion pour tout les messages au lieu d'en ouvrir et fermer à chaque envoie ce qui est mieux d'un point de vue temps.

5 Transmission d'objets

L'inconvénient du infrastructure n'offrant pas de validation de la syntaxe est que la forme de l'information doit être connue par les utilisateurs. En effet, on n'arrive pas à identifier la donnée par exemple avec des balises ou autre. Ceci est gênant si on crée une application et qu'elle doit récupérer des données d'un serveur, comme la structure n'est pas défini on c'est pas comment gérer l'information reçu. L'avantage est que se type format est léger car il ne comporte pas de balise comme en XML pour indiquer le nom de la donnée donc moins de caractère, c compréhensible et facile à utiliser, il n'a pas besoin de bien connaître la syntaxe pour l'utiliser.

6 Transmission comprimée

Le taux de compression dépend des données compresser. Si on compresser un fichier avec beaucoup de répétition, on aura un bon taux car on pourra ainsi joindre un maximum de données et enregistrer juste le nombre et le positionnement, ce qui est moins lourds. On peut en moyenne espérer un gain de 2 fois plus petit que le fichier original. Dans un texte, certaines lettres se répètent beaucoup de fois, comme par exemple le "e" et aussi plus le texte est grand plus cela vaut la peine parce qu'on pourra réduire efficacement la taille en joignant un maximum de caractère.