Systèmes mobiles Laboratoire n°2 : Protocoles applicatifs

Ce laboratoire propose d'illustrer l'utilisation de protocoles de communication applicatifs, basés sur http, dans le cadre d'applications mobiles

Questions

4.1 Traitement des erreurs

Si le serveur n'est pas joignable ou s'il retourne un code HTTP d'erreur, la requête tombe dans le vide et ne sera jamais traitée par le serveur. L'application ne recevra donc jamais de réponse. Il faudrait rajouter à SymComManager une méthode **onRequestFailure(String error)** et une méthode **handleServerError(String error)** à l'interface CommunicationEventListener

4.2 Authentification

Une authentification dans un protocole asynchrone est possible pour autant que le serveur et l'utilisateur se connaissent déjà. Une solution est d'utiliser une clé SSH, qu'il faut générer pour chaque utilisateur (appareil).

Une transmission différée est possible. On envoie les requêtes une fois que l'authentification est validée.

4.3 Threads concurrents

Cela peut poser un problème d'intégrité des données. Il faut s'assurer que les données lues et modifiées par les threads le soient de manière exclusive.

4.4 Ecriture différée

Une connexion par transmission différée a l'avantage de bien séparer les requêtes et donc leur traitement (par le serveur par l'application). Cependant, chaque connexion a un coût, peut surcharger le serveur et les requêtes peuvent mettre plus de temps à être traitées.

Multiplexer les connexions implique d'avoir un protocole bien établi entre le serveur et l'application, que la structure des messages soient bien définies pour pouvoir distinguer chaque transmission. Les requêtes peuvent être plus lourdes et demander plus de mémoire. L'avantage principal est qu'on utilise qu'une seule connexion, donc moins de risque de surcharge côté serveur, et moins de risque de connexion qui échoue.

On pourrait imaginer un mix des deux, selon le type de données qu'on voudrait envoyer. Certaines données se prêtent bien à être regroupées et compressées, d'autres moins. (Voir slides sur la sérialisation)

4.5 Transmission d'objets

a. L'avantage d'utiliser une infrastructure comme SOAP est de pouvoir valider les données reçues plus rigoureusement et de structurer les données. Cela facilite ensuite leur traitement.

L'avantage d'utiliser une infrastructure de type REST/JSON est la flexibilité. Il est plus facile d'évoluer notre environnement sur du JSON qu'avec du XML+DTD. On peut noter aussi qu'une sérialisation en JSON est généralement plus légère qu'en XML.

b. Protocol Buffer peut être compatible avec HTTP. En effet, HTTP se contente d'envoyer une suite de byte peu importe sa nature. Il faut, par contre, que l'application et le serveur soit d'accord sur la structure du Protocol Buffer.

L'avantage de Protocol Buffer est qu'il se prête bien à des descriptions d'objet (Java par exemple). Il est optimisé pour ça, les messages seront donc plus légers qu'avec un JSON ou XML. Moins flexible que JSON et moins portable que XML.

c. On pourrait imaginer un système qui fasse des requêtes plus précises, plus proches de ce que SQL peut offrir.

4.6 Transmission compressée

JSON de 151KB avec compression niveau 9 :

2605ms

2461ms

2303ms

JSON de 151KB sans compression :

12467ms

716ms

735ms

JSON de 3269KB avec compression niveau 9 :

75171ms

JSON de 3269KB sans compression :

19411ms

6471ms

5081ms

JSON de 3269KB avec compression niveau 1 :

74506ms

On remarque que les requêtes sont exécutées plus rapidement sans compression, alors que ça devrait être le contraire...

Tests effectués sur connexion mobile, il se peut donc que le débit n'ait pas été suffisamment constant pendant leurs réalisations.