SYM - Laboratoire 02 - Réponses aux questions

3.2 Transmission différée

Notre implémentation présente les limitations suivantes: Les choses à transmettre en différé sont seulement en mémoire et donc non persistés, si on quitte l'application on va perdre toutes ces requêtes différées en attente.

La meilleure manière de le faire est persister les données. Tout cela avec une bonne structure de données pour optimiser et organiser l'ordre des requêtes avec le parallélisme.

4 Questions

4.1 Traitement des erreurs

> Les classes et interfaces SymComManager et CommunicationEventListener, utilisées au point 3.1, restent très >>(et certainement trop) simples pour être utilisables dans une vraie application: que se passe-t-il si le serveur n'est pas joignable dans l'immédiat ou s'il retourne un code HTTP d'erreur? Veuillez proposer une nouvelle version, mieux adaptée,de ces deux classes / interfaces pour vous aider à illustrer votre réponse.

Il faut que le traitement des erreurs soit présent. On peut aussi rajouter un bon support des status HTTP.

Dans le cas où le serveur répond un statut d'erreur ou alors dans le cas d'une indisponibilité du réseau , notre application pourrait réessayer de faire la requête par exemple toute les 30 secondes (ou à un intervalle de n secondes).

4.2 Authentification

> Si une authentification par le serveur est requise, peut-on utiliser un protocole asynchrone ? Quelles seraient les restrictions ? Peut-on utiliser une transmission différée ?

Oui si l'application, qui fait la requête en question, stocke les credentials pour pouvoir s'identifier. Par exemple de nos jours JWT (Json Web Token) est un moyen populaire pour pouvoir s'identifier.

Par contre si l'utilisateur doit s'identifier à chaque fois alors c'est non la réponse.

4.3 Threads concurrents

> Lors de l'utilisation de protocoles asynchrones, c'est généralement deux threads différents qui se répartissent les différentes étapes (préparation, envoi, réception et traitement des données) de la communication. Quels problèmes cela peut-il poser ?

Généralement les problèmes sont les mêmes que lors en programmation multi-threadée. La synchronisation et coordination des threads, ainsi que l'échange des informations entre threads.

4.4 Écriture différée

> Lorsque l'on implémente l'écriture différée, il arrive que l'on ait soudainement plusieurs transmissions en attente qui deviennent possibles simultanément. Comment implémenter proprement cette situation ? Voici deux possibilités:
>
> - Effectuer une connexion par transmission différée
> - Multiplexer toutes les connexions vers un même serveur en une seule connexion de transport. Dans ce dernier cas, comment implémenter le protocole applicatif, quels avantages peut-on espérer de ce multiplexage, et surtout, comment doit-on planifier les réponses du serveur lorsque ces dernières s'avèrent nécessaires ?
>
> Comparer les deux techniques (et éventuellement d'autres que vous pourriez imaginer) et discuter des avantages et inconvénients respectifs.

L'avantage d'envoyer chaque requête individuellement est que en cas de coupure un peut simplement la refaire. Si on regroupe les requêtes similaires et compatibles en une grande requête on doit tout recommencer en cas de coupure. Cette solution e tout regrouper peut être intéressante au niveau de la batterie.

Les requêtes qui nécessitent une réponse peuvent être triées des autres et exécutées selon un ordre qui permet de le faire (par exemple file d'attente ou alors un graphe potentiel-tâche).

L'avantage du multiplexage est que on peut éventuellement gagner du temps avec du parallélisme, si le serveur le supporte.

Pour mettre en place le protocole, il peut être intéressant d'utiliser GraphQL, car il permet de choisir ce qu'on veut reçevoir. Ainsi c'est un bon outil pour combiner plusieurs requêtes compatibles en une.

4.5 Transmission d'objets

> a. Quel inconvénient y a-t-il à utiliser une infrastructure de type REST/JSON n'offrant aucun service de validation (DTD, XML-schéma, WSDL) par rapport à une infrastructure comme SOAP offrant ces possibilités ? Est-ce qu'il y a en revanche des avantages que vous pouvez citer ?

On peut soumettre des éléments qui ne respectent pas la forme acceptée par le serveur avec du JSON. La détéction et géstion d'erreurs devient plus complexe au niveau du serveur. Ce n'est pas le cas avec les options qui offrent la validation, car chaque erreur de forme est immédiatement détectée au tout début, lors de la validation.

L'avantage de JSON reste qu'il est peu verbeux, c'est à dire qu'il est de plus petite taille pour une requête donnée en comparaison des autres. Ainsi on surcharge moins le réseau.

> b. L'utilisation d'un mécanisme comme Protocol Buffers est-elle compatible avec une architecture basée sur HTTP ? Veuillez discuter des éventuelles avantages ou limitations par rapport à un protocole basé sur JSON ou XML ?

Voici quelques articles qui nous ont aidé à écrire notre réponse:

- https://proandroiddev.com/protobuf-in-android-55b01d855c40
- https://codeclimate.com/blog/choose-protocol-buffers/
- https://reasonablypolymorphic.com/blog/protos-are-wrong/

Oui c'est possible d'utiliser Protocol Buffers via Http. Mais seulement dans le cas ou le client et le serveur sont écrit dans les langages qui le supportent. Par exemple cela n'est pas une bonne idée si le client est un simple navigateur web.

De plus c'est plutôt assez lourd à mettre en place en comparaison de XML ou JSON. Apèrs cela est plus efficace au niveau du réseau. Pour finir on pense que pour des applications à petite et moyenne échelle, où le traffic réseau le plus optimal n'est pas une nécessité, l'utilisation de Protocol Buffers n'est pas une bonne idée.

> c. Par rapport à l'API GraphQL mise à disposition pour ce laboratoire. Avez-vous constaté des points qui pourraient être améliorés pour une utilisation mobile ? Veuillez en discuter, vous pouvez élargir votre réflexion à une problématique plus large que la manipulation effectuée.

Dans la query : `{"query" : "{allAuthors{id first_name last_name}}"}` un système de pagination pourrait être fait, car si la liste de tout les auteurs est tro importante, cela impliquera une perte de temps et de bande passante. Car il est bien plus judicieux de faire du lazy loading (c'est à dire charger que les éléments dont on a besoin au moment où on en a besoin, par exemple pour l'affichage).

4.6 Transmission compressée

> Quel gain de compression (en volume et en temps) peut-on constater en moyenne sur des fichiers texte (xml et json sont aussi du texte) en utilisant de la compression du point 3.4 ? Vous comparerez plusieurs tailles et types de contenu.

Nous n'avons pas pu réaliser cette partie à cause d'une erreur 500 du serveur. Mais de manière générale, plus le texte est long, plus le gain de temps en le compressant est grand, surtout avec une mauvaise connexion.