4 Questions

4.1 Authentification

Il est tout à fait possible de mettre en place un système d'authentification. On peut obliger les clients à envoyer une clé de session à l'intérieur du JSON de chaque requête. Ainsi, le serveur pourra vérifier si le client et authentifier et si sa session n'a pas expiré.
Avec une telle méthode, il faudrait mettre en place un petit formulaire de connexion permettant de demander au serveur une clé de session. Si le serveur valide la connexion du client, il lui renvoie une clé de session unique.

Une fois connecté, les requêtes ressemblerait à ceci:

```
{
    "session": "f32oif23ou09sudf0932uif98u2",
    "data": {
        // Some data
    }
}
```

4.2. Threads concurrents

Il faut faire attention aux sections critiques et aux variables partagées, s'il y en a.

Par exemple, lors de notre exemple d'authentification, il faudrait éviter que le client envoie une requête avant d'avoir reçu la clé de session.

4.3. Ecriture différée

Pour illustrer les problème possible, imaginons le cas extrême où le client n'a pas pu se connecter depuis plusieurs jours.

Si on effectue une connexion par transmission différée, on peut se retrouver dans une situation où on essaye d'ouvrir des milliers de connexions en parallèle, ce qui peut facilement faire

crasher le client ou le serveur.

À l'inverse, si on utilise une seule connexion, on risque d'essayer d'envoyer plusieurs Go de données, ce qui sera extrèmement lent et n'aboutira probablement pas.

L'idéal serait d'ouvrir un nombre limité de connexions en parallèle (par exemple 20), et d'envoyer les données par packet de 10 requêtes (par exemple). Ainsi, on évite de rendre le système trop instable et on assure une bonne connexion entre le client et le serveur.

4.4. Transmission d'objets

Le fait de n'avoir aucun système de vérification impose d'avoir une très bonne documentation et de vérifier soi-même les données reçues. C'est une source d'erreur et donc de crash potentiel.

L'avantage est que le JSON est relativement simple à utiliser et facile à mettre en place. De plus, parser du XML est fastidieux alors que parser du JSON est beaucoup plus facile.

4.5. Transmission comprimée

Le taux de compression sur un texte varie entre 70% et 80%. Étant donné que le JSON contient beaucoup de symboles similaires (`{` , `}` , `[` , `]` , `,` , `"` , `:`) le taux de compression sera probablement plus élevé que sur un texte standard.

(Source: http://www.maximumcompression.com/data/text.php)