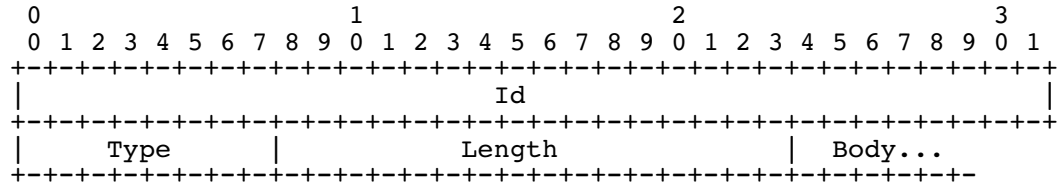
Protocoles Internet
TP 6 : UDP

Juliusz Chroboczek

13 novembre 2023

Exercice 1. Dans cet exercice, nous utilisons un protocole requête-réponse, où tous les datagrammes UDP auront la syntaxe suivante :



Le champ *Id* contient un identificateur qui permet d'apparier requêtes et réponses. Le client le choisit arbitrairement (il peut par exemple utiliser un compteur séquentiel), et le serveur le recopie depuis la requête vers la réponse. Le champ *Type* contient le type de message ; s'il est copris entre 0 et 127, le message est une requête, sinon c'est une réponse. Le champ *Length* contient la longueur du corps en octets. Le champ *Body* contient le corps du message, et il a une longueur de *Length* octets. Le corps du datagramme peut être suivi de données supplémentaires : si le corps du datagramme est plus long que $4 + 1 + 2 + \textit{Length}$ octets, alors les données supplémentaires sont ignorées [1].

Le protocole est strictement requête réponse : à chaque requête correspond exactement une réponse. On définit les requêtes suivantes :

- Type = 0 : *get-quotation*. Demande au serveur de retourner une citation littéraire tirée au hasard. Le corps doit être vide.

On définit les réponses suivantes :

- Type = 128 : *quotation*. Le corps est une citation littéraire codée en UTF-8.
- Type = 129 : *error*. Le corps est un message d'erreur codé en UTF-8.

1. Écrivez un programme qui décode l'objet JSON se trouvant à l'URL `https://jch.irif.fr:8443/udp-addresses.json` pour obtenir la liste des adresses de *socket* du serveur UDP.

1. Ce *footer* est prévu pour pouvoir contenir une signature cryptographique, mais nous ne l'utilisons pas dans ce TP.

2. Modifiez votre programme pour qu'il envoie une requête au serveur UDP et affiche le corps de la réponse reçue ; votre programme devra vérifier que l'*Id* reçu dans la réponse est égal à l'*Id* que vous avez émis. Que se passe-t-il si la requête ou la réponse sont perdues ?
3. Modifiez votre programme pour qu'il attende la réponse pendant deux secondes, puis reémette la requête si aucune réponse n'est reçue. (Vous pourrez utiliser la méthode `SetReadDeadline`.) Faut-il utiliser un *Id* différent à chaque fois, ou peut-on réutiliser le même *Id* pour les réémissions ?
4. Modifiez votre programme pour qu'il utilise l'algorithme du *backoff exponentiel* : la première réémission se fait au bout de 1 s, la deuxième au bout de 2 s, la troisième au bout de 4 s, etc. Que faut-il faire lorsque le temps de réémission a atteint une valeur déraisonnable (par exemple 64 s) ?

Exercice 2.

1. Modifiez le programme de l'exercice précédent pour que, à chaque fois qu'il a reçu une citation, il attende 5 s puis recommence.
2. Modifiez votre programme pour qu'il estime le RTT de chaque requête. Pour cela, vous maintiendrez deux variables :

$$\begin{aligned} \mathrm{RTT} &:= 2\,s \\ \mathrm{RTTvar} &:= 0\,s \end{aligned}$$

   et à chaque nouveal échantillon τ mesuré, vous ferez :

$$\begin{aligned} \delta &:= |\tau - \mathrm{RTT}| \\ \mathrm{RTT} &:= \alpha\,\mathrm{RTT} + (1-\alpha)\tau \\ \mathrm{RTTvar} &:= \beta\,\mathrm{RTTvar} + (1-\beta)\delta \end{aligned}$$

   Vous prendrez $\alpha = 7/8$ et $\beta = 3/4$.
   Si un paquet est perdu, il y a une ambigüité : la réponse correspond soit à la requête originale, soit à la requête reémise. Vous pourrez résoudre ce problème de deux manières :
   - soit en utilisant un *Id* différent pour chaque requête, même si c'est une reémission, ce qui permet de lever l'ambigüité ;
   - soit en considérant que la réponse correspond à la requête d'origine (en ne remettant pas l'horloge à zéro lors d'une réémission), ce qui revient à potentiellement sur-estimer le RTT (ce qui est moins grave que le sous-estimer).
3. Modifiez votre programme pour que le délai avant de faire une réémission soit égal à un RTO, où

$$\mathrm{RTO} = \mathrm{RTT} + 4\,\mathrm{RTTvar}.$$

4. Le serveur fourni n'a qu'une adresse. Que faudrait-il changer si le serveur annonçait plusieurs adresses ?