Second TD Client-Serveur RPC

Première partie

Elle consiste à étudier au moyen de traces le fonctionne du mode où le client et le serveur n'utilise pas le portmapper. Elle permet aussi d'exécuter un client et un serveur avec le transport UDP.

1. Vous allez utiliser l'archive *DictNoPortmapNoName.tar.gz*. Vous la désarchivez et tapez la commande *make* pour obtenir le serveur *rdictdemon* (sans contacter le portmapper) et le client *rdict* qui contacte directement le serveur. Lancez uniquement le serveur. Donnez le port *XXX* qu'utilise votre serveur au moyen de *netstat -tlp*.
2. Vous lancez une capture avec *wireshark* sur l'interface de rebouclage *lo* et exécutez le client pour qu'il utilise l'interface de rebouclage et le port *XXX*. Donnez la commande pour le client.
3. Arrêtez la capture dès que vous voyez une demande de connexion entre le port de votre serveur RPC et le client. Donnez le port du client et faites une copie d'écran des trois segments TCP de demande d'établissement de connexion.
4. Vous modifiez maintenant le client *rdict.c* pour qu'il utilise UDP via *clntudp_create*. Vous expliquez les différents paramètres de cette fonction en utilisant une variable *struct timeval timeout* avec un délai de 3 secondes.
5. Vous compilez et exécutez votre client en vous aidant de la commande *netstat -ulp*. Vous expliquez pourquoi votre serveur fonctionne en UDP.
6. Vous allez modifier votre serveur pour qu'il ne réponde qu'à des requêtes en UDP. Vous expliquez vos modifications. Vous exécutez votre serveur. Vous vérifiez avec votre client UDP.
7. Vous donnez la commande qui permet d'être sûr que le serveur n'accepte plus les demandes de connexion TCP.

Seconde partie

Elle vise à modifier la spécification, c'est à dire l'interface des procédures, partagée par le client et le serveur. Nous allons aussi étudier en détail les fichiers générés.

1. Vous allez ajouter un paramètre *DictName* à la procédure *initdict()* qui correspond à une structure *DictName* similaire à *DictRecord* c'est à dire avec un champ *longueur* et *donnee*. La donnée permettra de transmettre le nom de l'utilisateur du dictionnaire. Modifiez le fichier *rdict.x* en conséquence.
2. Compilez en exécutant *make proper* puis *make rpc*. Vérifiez avec les dates quels sont les fichiers générés ?
3. Vérifiez dans *rdict.h* que la modification pour les types des structures et la procédure *initdict* a bien eu lieu ?
4. Vérifiez pour le fichier *rdict_clnt.c* que la modification a eu lieu. Commentez le rôle de la fonction *clnt_call* appelée et ses différents paramètres.
5. Vérifiez pour le fichier *rdict_svc.c* la modification. Commentez le code exécuté lorsqu'une requête *INIDICT* parvient au serveur. Vous expliquez pourquoi la procédure *rdictprog_1* est exécutée. Vous commentez chacune des lignes de code exécutée dans le cas d'une telle requête notamment la ligne *result = (*local)((char *)&argument, rqstp)* et l'appel à *svc_sendreply*.
6. Compilez le client via *make rdict*.

7 Assurez vous que le code métier appelle bien la fonction *initdict* avec le nom saisi par l'utilisateur. Compilez à nouveau votre client via *make rdict*.
8. Compilez le serveur via *make rdictdemon* en vous assurant qu'il n'utilise pas le portmapper.
9. Modifiez le code métier du serveur pour qu'il affiche lors de *initdict* le nom reçu du client dans le paramètre *DictName*. Compilez à nouveau via *make rdictdemon*.
10. Exécutez votre serveur. Vérifiez qu'il reçoit bien le nom saisi et faites une capture *wireshark* pour confirmer que le nom a bien circulé entre le client et le serveur. Donnez le copie d'écran de cette capture.