

Projet

Aidez les libraires !

1 Description


En cette période troublée pour les commerces de proximité, vous souhaitez venir en aide aux petites librairies indépendantes. Pour cela, on propose de rédiger un système baptisé « Nil¹ » pour collecter les demandes des clients et les orienter vers le ou les libraires chez qui ils peuvent commander les livres et les chercher en mode « click and collect ».

Lorsqu'un client interroge le serveur Nil, il indique les références du ou des ouvrages souhaités. Nil interroge alors les librairies adhérentes pour déterminer lesquelles disposent des ouvrages demandés, puis il transmet les adresses des librairies concernées aux clients. Ces derniers peuvent alors réserver les livres directement auprès des librairies.

Le problème est que les librairies peuvent être fermées (ou en dérangement, ou déconnectées du réseau, etc.) au moment où Nil les interroge. Il faut donc traiter les requêtes avec une certaine tolérance aux pannes. Les clients, quant à eux, souhaitent une réponse rapide de Nil sinon ils risquent d'aller voir ailleurs.

2 Travail demandé

On demande de rédiger les programmes suivants :

— `librairie port livre1 livre2 ... livren` 

Ce programme simule une librairie : son stock initial est indiqué par les références des livres fournies sur la ligne de commande. Une référence peut être citée plusieurs fois si le stock contient plusieurs exemplaires du livre. Pour simplifier, on considérera qu'une référence de livre est une chaîne d'au plus 10 caractères.

La librairie attend des requêtes du serveur Nil formulées avec UDP ainsi que des requêtes des clients formulées avec TCP pour réserver un ou plusieurs ouvrages. Dans les deux cas, la librairie attend sur le port indiqué (le même en TCP et en UDP pour simplifier).

— `nil port délai librairie port librairie port...`

Ce programme est le serveur Nil au cœur du système : chaque librairie est indiquée par son adresse (nom ou adresse IPv4 ou IPv6) et son numéro de port UDP. Le serveur attend des requêtes des clients, formulées avec TCP sur le port indiqué en premier argument. Dès qu'il reçoit une requête, il interroge toutes les librairies connues en utilisant UDP.

Du fait des contraintes du commerce en ligne, le délai maximum d'attente du client doit être borné par l'argument *délai* (en secondes, typiquement de l'ordre de 1 à 10 secondes), ce qui implique que le serveur doit interroger toutes les librairies en parallèle : le serveur lance les requêtes vers les librairies les unes après les autres, mais n'attend pas la réponse avant d'interroger la librairie suivante. Si toutes les librairies ont répondu avant l'expiration du délai, la réponse est envoyée au client. Si une ou plusieurs librairies n'ont pas transmis de réponse à l'expiration du délai, le serveur renvoie au client les informations en sa possession à ce moment. **C'est ce mode de fonctionnement avec un délai borné et la tolérance aux pannes qui impose d'utiliser UDP.**

Les informations renvoyées par le serveur au client sont, pour chaque ouvrage demandé, l'adresse (IPv4 ou v6) et le numéro de port de la librairie possédant l'ouvrage dans son stock. On notera que le serveur peut recevoir des requêtes de plusieurs clients simultanément, ce qui impose que le serveur sache distinguer les réponses parvenant des librairies pour chaque client.

— `client serveur port livre1 livre2 ... livren`

Ce programme simule un client : il interroge le **serveur indiqué** indifféremment par son nom, son adresse IPv4 ou IPv6 ainsi que par son numéro de port TCP. Une fois les réponses obtenues, le client contacte avec TCP la ou les librairies indiquées pour réserver les livres.

1. Le Nil est l'autre plus grand fleuve du monde.

3 Exemple de session



On trouvera ci-après un exemple (fictif) d'utilisation des trois programmes. Les affichages des différents programmes dans cet exemple ne sont là que pour expliciter les étapes et donner des idées pour vos propres affichages.



```
> ./nil 9000 5 127.0.0.1 9001 \
# le serveur écoute sur le port 9000
Recu requete de ::1/1234 pour A B X Z
Recu reponse de 127.0.0.1/9001
Recu reponse de ::1/9003
Envoi reponse au client
# réponse envoyée avant 5 secondes
^C

> ./librairie 9003 A A B C D
# cette librairie a 5 livres en stock
Recu requete 65535 de ::1/9000
Envoi reponse 65535 A B
Cmd recue de ::1/1235 : A B
Stock restant : A C D
^C

> ./client ::1 9000 A B X Z
Envoi requete a ::1/9000
A : dispo sur 127.0.0.1/9000, je commande
B : dispo sur 127.0.0.1/9000, je commande
X : dispo sur 130.79.255.66/9001, je commande
Z : pas dispo
# terminé
```

4 Spécification des protocoles

Cette section décrit les protocoles que vous devez implémenter dans le cadre de ce projet.

4.1 Protocole serveur Nil ↔ librairie

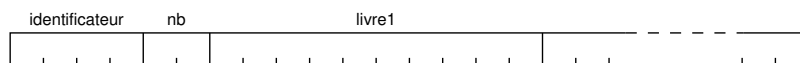


FIGURE 1 – Protocole entre le serveur Nil et les librairies

Le protocole utilisé par le serveur Nil pour obtenir la disponibilité des livres dans les librairies repose sur UDP et est représenté sur la figure 1. Lorsque le serveur souhaite obtenir la disponibilité d'une série de livres, il envoie à chaque librairie une requête dans un unique datagramme comportant un identificateur (sur 4 octets, compris entre 0 et $2^{32} - 1$) permettant de distinguer les différentes demandes des clients en cours de traitement, le nombre de livres demandés (sur 2 octets, compris entre 0 et 65535) ainsi que les références bibliographiques souhaitées, chacune prenant exactement 10 octets. Si une référence fait moins de 10 octets, on complétera avec des octets nuls.

La réponse faite par la librairie suit le même format, mais n'indique que les références trouvées dans le stock. Si aucune référence n'est trouvée, la librairie envoie une réponse indiquant un nombre nul de références.

On supposera (sans le vérifier) que les messages échangés tiennent tous dans un datagramme. On notera en outre qu'il s'agit d'un protocole sans état : la librairie ne conserve aucune information sur le serveur.

Suggestion d'extension optionnelle : pour accroître la tolérance aux pannes et pour éviter qu'une perte de datagramme ne se traduise par une vente ratée, le serveur Nil peut émettre une nouvelle requête à destination d'une librairie s'il n'a pas reçu de réponse après un certain délai, comme par exemple à la moitié du délai maximum imposé au serveur.

4.2 Protocole client ↔ serveur Nil

Le protocole utilisé par un client pour interroger le serveur Nil repose sur TCP et est représenté sur la figure 2. Le client initie une connexion TCP sur le serveur et lui envoie les références bibliographiques souhaitées, suivant un format comparable à celui du protocole précédent, mais sans l'identificateur.

Au bout du délai maximum fixé pour le serveur, ou plus tôt si celui-ci a reçu suffisamment de réponses, le serveur répond en donnant le nombre d'éléments trouvés (sur 2 octets, entre 0 et 65535) puis en fournissant chaque élément sous forme d'une suite de 29 octets comprenant 10 octets pour la référence de l'ouvrage, un octet valant 4 ou 6 pour indiquer si l'adresse IP de la librairie est une adresse IPv4 ou IPv6, puis l'adresse elle-même sur 16 octets (s'il s'agit d'une adresse IPv4, cela signifie que 12 octets ne contiendront aucune information utile), et enfin 2 octets pour le port TCP de la librairie.

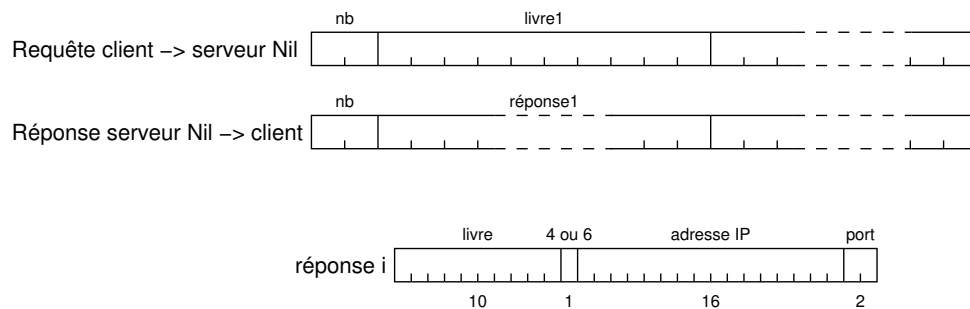


FIGURE 2 – Protocole entre les clients et le serveur Nil

Le serveur Nil doit pouvoir gérer les requêtes de plusieurs clients en parallèle.

4.3 Protocole client ↔ librairie

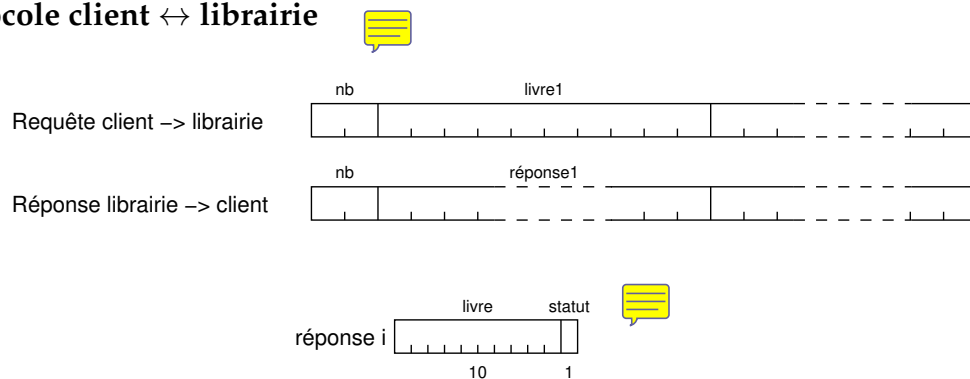


FIGURE 3 – Protocole entre les clients et les librairies

Lorsque le client a reçu la réponse du serveur Nil, il connaît les librairies où commander chaque livre trouvé et peut donc réserver les ouvrages en utilisant TCP et le protocole représenté sur la figure 3. Le client initie une connexion TCP vers la librairie et lui envoie les références bibliographiques souhaitées dans un format comparable au protocole utilisé avec le serveur Nil.

En retour, la librairie répond avec pour chaque livre le statut de la réservation sur un octet : 1 si la réservation est effective, l'ouvrage disparaît alors du stock de la librairie, ou 0 si la réservation ne peut être faite, comme par exemple si l'ouvrage a été réservé entretemps par un autre client.

5 Travail demandé

Faire une fonction main pour tester tout, et demande à l'utilisateur de remplir ses librairies, etc...

Ce projet est à réaliser individuellement. Vous devez implémenter les programmes mentionnés, en respectant scrupuleusement les protocoles définis ci-dessus. Vos programmes doivent être rédigés en langage C à l'aide des *sockets*, sans utiliser d'autre mécanisme ou bibliothèque de programmation réseau. Ils doivent être compilables sur la machine de référence `turing.unistra.fr` avec les options `-Wall`, `-Wextra` et `-Werror`, et exécutables sur cette même machine.

Vous rédigerez un rapport décrivant votre implémentation, les structures de données mises en œuvre, et les limitations et les extensions éventuelles que vous aurez réalisées.

Vous déposerez votre projet (documentation, sources, Makefile) sous forme d'une archive au format « `tar.gz` » sur Moodle, dans l'espace de devoirs prévu à cet effet. Vous prendrez soin de supprimer tous les fichiers binaires (exécutables, objets).