UE21 – Examen pratique de réseau

Enoncé de 2ème session – Septembre 2023
Année académique 2022-2023

Rappel

Cet examen **est personnel** et se déroule **à cours ouvert**. Vous pouvez disposer de tous vos documents, extraits de codes et consulter des sources internet. Je vous rappelle qu'un code est semblable à une rédaction : sa composition et sa structuration sont très personnelles. Ainsi, toute copie de code ou échange avec l'un de vos camarades sera sanctionné par la procédure disciplinaire en vigueur.

Enoncé

On vous demande d'écrire, en Java, un **programme serveur** permettant la gestion simplifiée d'un serveur de licences. Précisément, ce système doit permettre d'obtenir et de libérer des licences valides (ie. dont la date d'expiration n'a pas encore été atteinte) et de maintenir le stock. Dans ce système simplifié, seul le serveur sera développé et il annoncera les programmes disponibles en multicast.

Description technique

Au commencement, le programme serveur *envoie des annonces pour chaque programme pour lequel il propose des licences* (il peut y avoir plusieurs instances du programme serveur). Ces annonces sont transmises toutes les `30` secondes, en multicast.

Par ces annonces, les éventuels clients (qui ne doivent pas être développés ici) identifient les programmes pour lesquels des licences peuvent être obtenues. Ensuite, le client, en *se connectant en unicast au serveur*, **peut envoyer une demande de plusieurs licences d'un programme** et, en fonction des licences disponibles, le serveur répond favorablement ou non. Il peut également **libérer les licences allouées** (qui redeviennent alors disponibles pour d'autres clients). Le client (s'il est administrateur, mais nous ne vérifierons pas les rôles dans cette implémentation simplifiée) peut **ajouter des nouveaux programmes et licences** qui seront alors distribués par le serveur.

Grammaire et explication

La grammaire prévue pour ce protocole est rudimentaire. Notons que tous les messages se terminent par un retour à la ligne (`CR LF`) et que les chevrons < et > servent à désigner un paramètre (ne **sont pas** envoyés).

Ainsi les messages suivants sont prévus :

1. Échanges multicast (récurrence : `30` secondes, port : `60321`, IP : `226.225.224.224`)
   `LIC <program-id> <unicast-port>`

   Ce message `LIC` est envoyé en multicast. Il est transmis à l'adresse IPv4 `226.225.224.224`. Il permet à un client de connaître l'existence des programmes disponibles et des serveurs gérant les licences. Ce message mentionne l'identifiant d'un programme (`<program-id>`) et le port unicast (`<unicast-port>`) à utiliser pour consommer, libérer ou ajouter une licence sur ce serveur. Un message `LIC` distinct est envoyé pour chaque programme dont des licences sont proposées. Pour simplifier, nous supposerons qu'un programme donné n'est proposé que par un seul serveur de licences.

   Exemple avec les identifiants *programme1* et *programme2* distribué sur le port unicast `8372` :
   ```
   LIC programme1 8372
   LIC programme2 8372
   ```

   Le serveur peut être contacté par un client sur son adresse IP, sur le port TCP `8372`.

2. Échanges (unicast) direct avec le serveur (port : voir annonce `LIC`, **protégés par TLS 1.3**)
   `ASK <program-id> <license-count>`

```
ASKOK <license-id>
ASKERR
FREE <license-id>
FREEOK
FREEERR
ADD <program-id> <license-count> <expiration-date>
ADDOK
ADDERR
```

Le message « ASK » permet de demander la consommation d'un nombre déterminé (<license-count>) de licences pour le programme identifié (<program-id>). Le serveur répondra par un message « ASKOK » (mentionnant l'identifiant unique du groupe de licences <license-id>) si la demande peut être satisfaite ou par un message « ASKERR » dans le cas contraire.

Lorsque des licences sont octroyées, le serveur crée un identifiant (<license-id>) pour ce groupe. Il est composé ainsi : <program-id>-<num-id> (avec <num-id> un numéro de séquence, débutant à 1 et incrémenté à chaque octroi d'un groupe de licences).

Le message « FREE » permet de demander la libération d'un groupe de licences au serveur ayant octroyé celle-ci en mentionnant l'identifiant de ce groupe (<license-id>). S'il peut être libéré, le serveur répondra par un message « FREEOK », sinon un message « FREEERR » sera envoyé.

Le message « ADD » permet soit d'ajouter un nouveau programme servi par le serveur de licences ou alors, si le programme est déjà connu, d'ajouter des licences pour ce programme. Si le programme est déjà connu, les licences sont ajoutées au nombre déjà disponible et la date d'expiration est mise à jour. Si le programme n'est pas encore connu, il est ajouté à la liste des programmes proposés (et donc une nouvelle annonce LIC sera envoyée) par ce serveur. **Notons qu'il faut sauvegarder les données dans le fichier JSON à chaque commande ADD**.

Exemple d'échanges :

```
ASK programme1 5
ASKOK programme1-00001          ← Identifie le groupe de licences
FREE programme1-00001           ← Libère le groupe de licences (5)
FREEOK
FREE programme1-00001
FREEERR                         ← Ce groupe de licences n'existe plus (déjà libéré)
ADD programme5 3 13/12/2023
ADDOK
```

Comme mentionné, un **fichier JSON** est utilisé pour *persister les informations* concernant les licences proposées par le serveur. L'état du système (ie. les groupes de licences) n'est pas sauvegardé. Le squelette fourni propose déjà *le parseur* et *la gestion du fichier JSON*. L'outil MulticastSpy permet de capturer les échanges multicast.

Principales tâches à faire

1. Envoi des annonces LIC (en multicast, toutes les 30 s, port : 60321, IP : 226.225.224.224). Une annonce distincte est envoyée par programme disponible sur ce serveur.
2. Écouter sur le port unicast annoncé et, pour chaque client se connectant :
    a. Répondre aux requêtes ASK (consommation de licences) par ASKOK ou ASKERR.
    b. Répondre aux requêtes FREE (libération de licences) par FREEOK ou FREEERR.
    c. Répondre aux requêtes ADD (ajout de programme ou licences) par ADDOK ou ADDERR.

La possibilité de connexions simultanées implique de la programmation concurrente (ie. Threads). Veillez à protéger l'accès et l'utilisation des ressources communes.

Les échanges unicast doivent être protégés par TLS 1.3 avec le certificat fourni. Pour tester votre connexion, vous pouvez simuler un client avec l'outil OpenSSL (supportant TLS). Pour tester, il faut vous rendre dans le

dossier où l'outil a été installé et entrer la commande dans la console (pour une connexion sur `localhost` avec le port TCP `8372`, par exemple) :

```
openssl s_client -connect localhost:8372
```

ATTENTION ! Pour des raisons d'incompatibilité, entrez les messages en minuscule avec l'outil OpenSSL. L'analyseur (ie. la classe `Parser`) a été adapté en conséquence. Pour simuler plusieurs clients, démarrez plusieurs instances de l'outil dans des consoles différentes.

Grammaire ABNF complète

```
digit = "0"/ "1"/ "2"/ "3"/ "4"/ "5"/ "6"/ "7"/ "8"/ "9" ; un chiffre
letter = %x41-5A / %x61-7A ; une lettre majuscule ou minuscule
digit_letter = letter / digit
digit_letter_minus = letter / digit / "-"
space = " "
crlf = %x0D %x0A ; CR (code ASCII 13) suivi de LF (code ASCII 10)
programid = 3*15digit_letter
licenseid = 1*30digit_letter_minus
date = 2digit "/" 2digit "/" 4digit ; JJ/MM/AAAA
licensecount = 1*3digit
unicastport = 1*5digit
req = "LIC" space programid space unicastport crlf
ask = "ASK" space programid space licensecount crlf
free = "FREE" space licenseid crlf
askok = "ASKOK" space licenseid crlf
askerr = "ASKERR" crlf
freeok = "FREEOK" crlf
freeerr = "FREEERR" crlf
add = "ADD" space programid space licensecount space date crlf
addok "ADDOK" crlf
adderr = "ADDERR" crlf
```

Informations pratiques

- Documentation pour gérer TLS en Java avec *les keystore* et *truststore* : https://stackoverflow.com/questions/18787419/ssl-socket-connection
- L'outil OpenSSL est disponible sur tous les systèmes. Il est fourni en standard pour Linux ou MacOS. Pour Windows, il est disponible ici : https://swi.la/openssl
- La librairie GSON doit être utilisée pour gérer le fichier de données JSON (implémentation faite). Un fichier d'exemple est inclus dans le dossier `data`.
- Le certificat à utiliser est dans le dossier `cert` et le mot de passe est `cert2023`.

Niveau	Description	Note max
1	Implémentation des éléments réseaux : • *Echanges unicast (multi-utilisateur, non chiffré) : requêtes et réponses* • *Echanges multicast (non chiffré)* • *Stabilité du serveur & Respect des principes SOLID*	**15**
2	Gestion sécurisée des données et protection des ressources • *Protection des données partagées (accès concurrents…)* • *Gestion correcte des ressources allouées (allocation, accès, libération…)*	**18**
3	Implémentation du chiffrement TLS 1.3 • *Support TLS dans les échanges unicast*	**20**