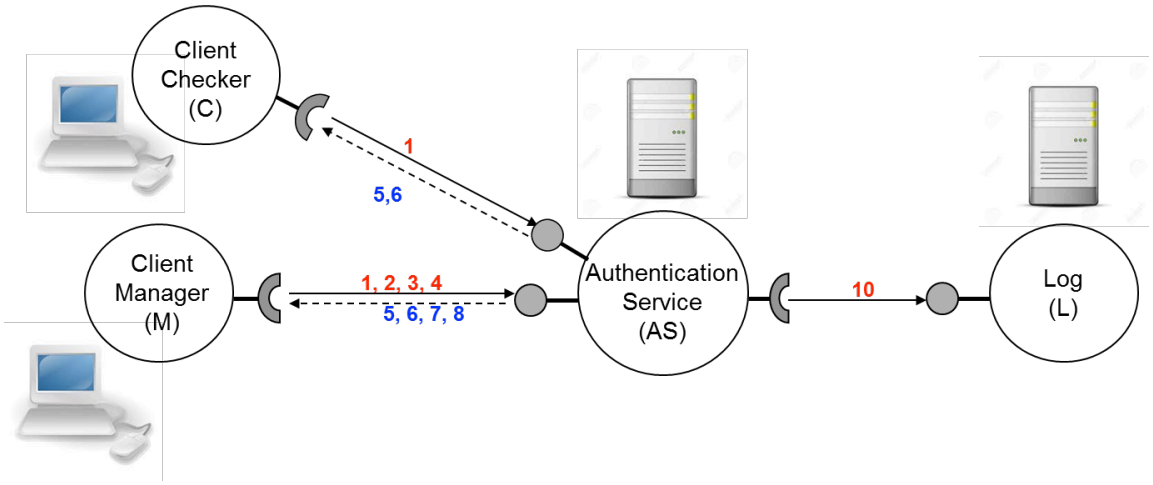
TP
Programmation API JAVA Sockets TCP/IP

On se propose de développer, de façon incrémentale, un service simplifié d'authentification basé sur des associations « *login*/*password* ». La figure ci-dessous illustre l'objectif fonctionnel à atteindre.



Le service d'authentification (AS) gère localement de façon persistante une liste de paires « *login*/*password* ». Deux types de processus distants peuvent lui demander la réalisation d'opérations sur cette liste. Ils occupent un rôle de client de ce service.

Pour une paire « *login*/*password* » fournie en entrée par l'utilisateur, un processus client Checker (C) permet d'en vérifier la validité auprès du serveur AS distant. Le second type de processus client, désigné par Manager (M) s'adresse à un utilisateur administrateur. En plus de la fonctionnalité de vérification de validité, il permet de demander à l'AS l'ajout ou la suppression d'une paire ainsi que la modification du *password* associé à un *login* au sein d'une paire.

Pour cela, on adopte les spécifications protocolaires (sémantiques et syntaxiques) suivantes :

Messages d'un client (C ou M) vers le serveur AS		Numéro sur figure	Type(s) Client concerné(s)
Requête de vérification d'une paire	**CHK** *Login Password*	**1**	**C et M**
Requête d'ajout d'une paire	**ADD** *Login Password*	**2**	**M**
Requête de suppression d'une paire	**DEL** *Login Password*	**3**	**M**
Requête de modification d'une paire	**MOD** *Login NewPassword*	**4**	**M**
Messages du serveur AS vers un client (C ou M)			
Réponse à vérification	**GOOD ou BAD**	**5 ou 6**	**C et M**
Compte rendu positif d'opération (ADD, DEL ou MAJ)	**DONE**	**7**	**M**
Compte rendu négatif	**ERROR** *Raison*	**8**	**M**

Exemples : CHK desprats hjl7-67tru
DEL..teyssie rcz!ou3008
ERROR name_unknown
MOD lavinal 23krest?l97
ADD torguet çrg89pom:15

Enfin, toute requête traitée par l'AS doit être journalisée auprès d'un service dédié dénommé Log (L). En plus de la requête elle-même, sont également enregistrés, l'horodatage correspondant ainsi que l'identifiant du client demandeur. Par identifiant, on sous-entend le triplet constitué par l'adresse IP de la machine sur laquelle s'exécute le processus client, le numéro de port de Transport utilisé par le processus client, ainsi que le protocole utilisé pour transporter la requête (TCP ou UDP). La composition des messages numérotés 10 sera détaillée par la suite.

Etape 1 : Code « métier » du service
Dans une étape préparatoire, récupérer la classe `ListeAuth.java` qui, au-delà du constructeur public `ListeAuth(String nomfichier)`, offre les méthodes permettant la réalisation des opérations de vérification, d'ajout, de suppression et de modification d'une paire « *login/password* » à partir de la liste rendue localement persistant dans le fichier texte `nomfichier`.

Etape 2 : Développement du service AS **version 1.a** : (C <- - - > AS) sur UDP
On suppose qu'un processus serveur AS réserve le port UDP 28414 aux interactions entre lui-même et les processus clients « Checker » (C).
Programmer en JAVA :

i. Le rôle d'un serveur AS limité à la seule opération de vérification
ii. Le rôle d'un client C

Etape 3 : Développement du service AS **version 1.b** : (C <- - - > AS) sur TCP
On suppose qu'un processus serveur AS réserve le port TCP 28414 aux interactions entre lui-même et les processus clients « Checker » (C).
Programmer en JAVA :

i. Le rôle d'un serveur AS limité à la seule opération de vérification
ii. Le rôle d'un client C

Etape 4 : Développement du service AS **version 1.0** : (C < - - - > AS) sur UDP et TCP
Il s'agit d'intégrer, pour le rôle de serveur AS, les versions 1.a et 1.b en une seule version « multi-protocole » : à l'issue de cette étape un processus serveur AS est capable de traiter des requêtes émanant d'un client distant « Checker » que celles-ci aient été transportées via UDP ou via TCP.

Etape 5 : Développement du service AS **version 1.c** : (M < - - - > AS) sur TCP
On suppose qu'un processus serveur AS réserve le port TCP 28415 aux interactions entre lui-même et les processus clients « Manager » (M).
Programmer en JAVA :

i. Le rôle d'un serveur AS
ii. Le rôle d'un client M

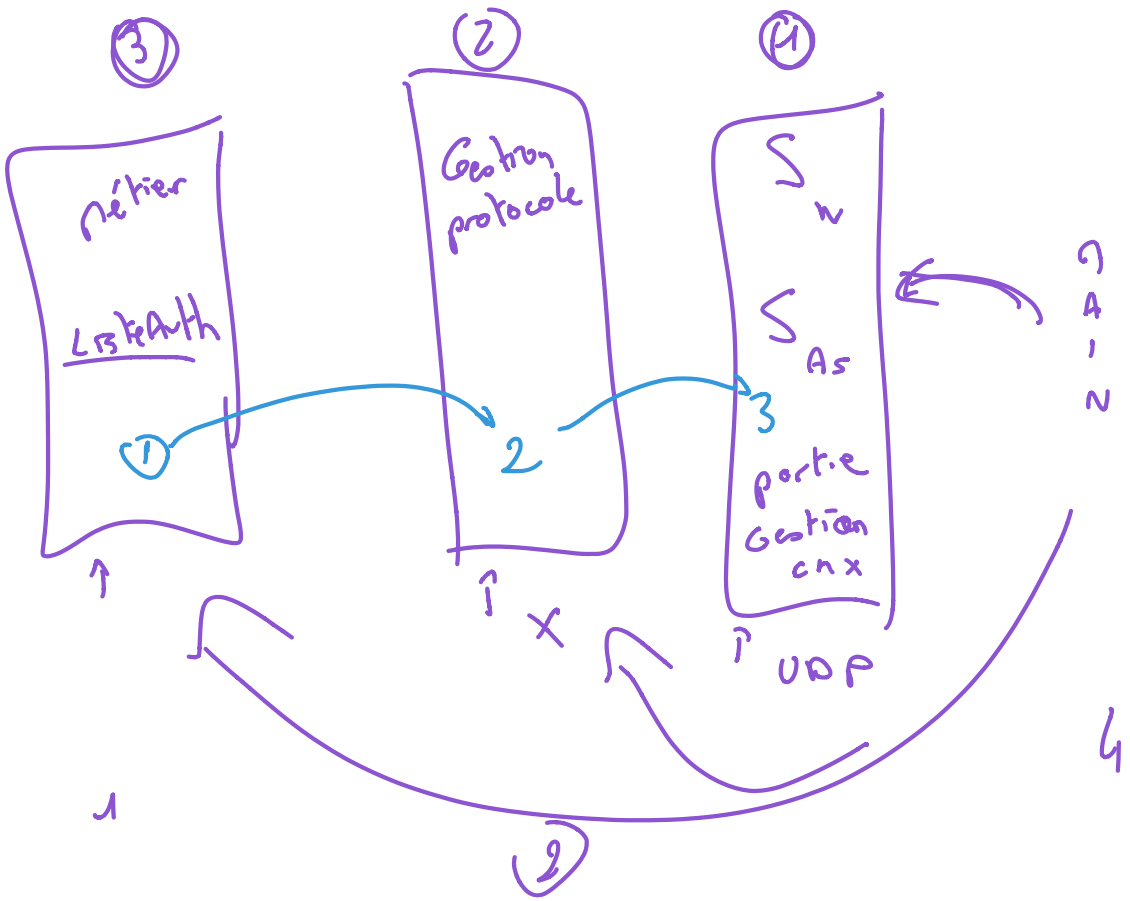
Etape 6 : Développement du service AS **version 2.0** : (C, M < - - - > AS)
Il s'agit d'intégrer, pour le rôle de serveur AS, les versions 1.0 et 1.c en une seule version : à l'issue de cette étape un processus serveur AS 2.0 est capable de traiter des requêtes émanant d'un client distant « Checker » que celles-ci aient été transportées via UDP ou via TCP, et celles émanant d'un client distant « Manager ».

Etape 7 : Représentation en JSON des messages de journalisation
On souhaite utiliser JSON pour le codage des requêtes d'enregistrement d'opérations. Afin de pouvoir faciliter les tests en mode réparti de l'échange des messages de journalisation (numérotés 10) entre vos solution, récupérez la proposition de représentation JSON intégrant toutes les données véhiculées.

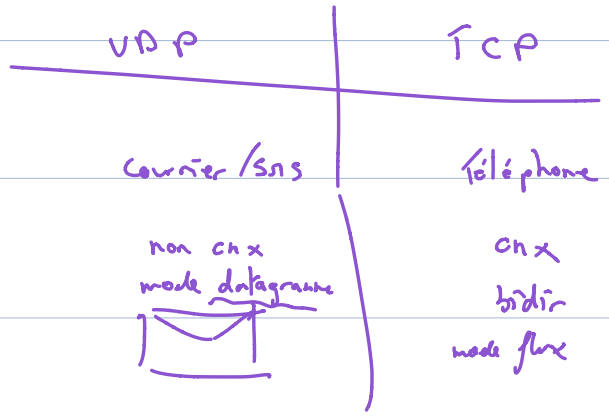
Etape 8 : Développement du service de journalisation (L)
On suppose qu'un processus serveur Log AS réserve le port TCP 3244 aux interactions entre lui-même et ses processus clients. Ici le développement se restreint aux interactions entre le client AS et le serveur L.
Programmer en JAVA :

i. Le rôle d'un serveur L
ii. Le rôle d'un client AS

Etape 9 : Développement du service AS version 3.0
Il s'agit d'intégrer au sein de la version 2.0 de AS le rôle de client du service de journalisation



UDP :



UDP :
- a) recevoir
  - prep : Socket UDP, 1L
  - 1) Recevoir 1L
  - 2) Extraire infos ~32L
- b) Traiter : 1L
- c) envoi
  - 1) Paquet (@ +port + contenu) Req / Rep 2L
  - 2) envoyer 1L

travail

```
prep.   DatagramSocket ds = new DS( port );
1       DatagramPacket p = new DP( char[128] );
        ds.receive(p);
2       req = p.getData();
        rep = gp.traiter(req);
        DP p2 = new DP(IPc, Portc, rep);
        ds.send(p2);
        ds.close();
```