mickael.hoerdt@hesge.ch 2015/2016

# Mini-projet d'ingénierie de protocoles réseaux informatiques

## *Un jeu de snake multi-joueur en réseaux*

### *Etape 1: le protocole SnakeChannel*

Le jeu du snake multi-joueur sera basée sur une architecture client-serveur et devra être écrit en python 2. Toutes les communications entre le client et le serveur seront basées sur le protocole SnakeChannel défini tout au long des étapes du projet. Ce protocole sera transporté au dessus d'UDP et d'IPv4 : cela signifie qu'il faudra obligatoirement utiliser des sockets de type SOCK_DGRAM.

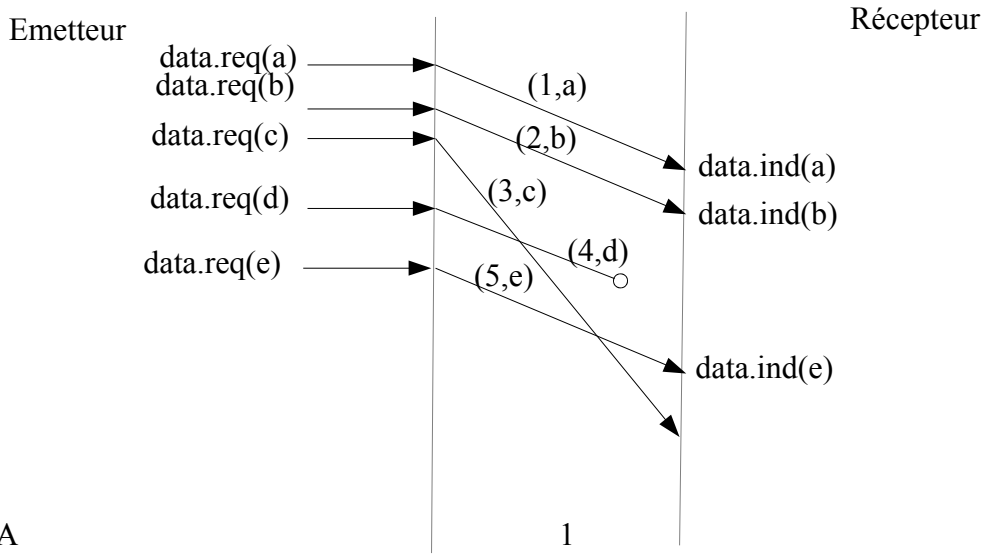
Les messages qui circulent entre les clients et le serveur, ainsi qu'entre le serveur et le/les clients contiendront les différents états du jeu. Ceux-ci peuvent se perdre, mais doivent obligatoirement arriver dans l'ordre, afin d'éviter que le serveur ou le/les clients traitent un état de jeu qui serait de toute façon périmé. La première étape de ce projet est la conception et l'implémentation d'un protocole **connecté** et **bidirectionnel** répondant à ses contraintes qu'on appellera le protocole *SnakeChannel*.

**La phase de circulation des données**

Ce protocole rajoute un numéro de séquence de 4 bytes devant les données à transporter utilisé pour filtrer les messages UDP qui seraient dupliqués ou qui n'arriveraient pas dans l'ordre croissant à l'intérieur du canal de communication.

Ces 4 bytes contiennent un numéro de séquence qui est incrémenté à chaque nouvelle transmission dans le SnakeChannel. A la réception, le numéro de séquence du paquet est comparé avec le dernier numéro de séquence reçu, si celui-ci ne lui est pas supérieur, le paquet est sera ignoré.

Le diagramme temps séquence ci-dessous illustre un exemple d'exécution du protocole : Le message **a** arrive et contient un numéro de séquence supérieur à attendu par le récepteur, le message **c** est déséquencé et ne contient pas un numéro de séquence supérieur à celui attendu et est donc ignoré. Le message **e** est plus récent que tous les autres reçu jusqu'à présent et est par conséquent envoyé à l'application.



**La phase de connexion**

Avant de pouvoir envoyer des données sur SnakeChannel, il faut préalablement s'y connecter. Pour cela, le protocole SnakeChannel dispose de messages *hors-bande* identifiés par un numéro de séquence égal à `0xffffffff`.

Comme les messages peuvent se perdre, cette connexion a lieu en 4 phases. Si un message du coté client ne reçoit pas de réponse au bout temps fixe, il doit être renvoyé. Le serveur ne fait que répondre au message venant d'un client.

1- La première phase est l'envoie d'une chaine ascii *« GetToken» A Snake* d'un nouveau client vers le serveur où *A* est un nombre entre 0 et 2^32-1 choisi au hasard.

2- Si ce message arrive, le Serveur réponds par un message ascii *«Token» A B Pnum* où A est un nombre entre 0 et 2^32-1 choisi au hasard, *B* la valeur reçu par le client dans le message *GetToken* et *Pnum* le numéro du protocole utilisé par ce serveur

3- Le client récupère le numéro de token du serveur, et renvoie un message ascii *«Connect»* suivi d'une chaine de caractère encodant des clés/valeurs dont le format est le suivant :

```
"\nom_clé\valeur_clé\nom_clé\valeur_clé\...\..."
```

Les nom des clefs et leur valeurs qui doivent obligatoirement être présentes dans la réponse du client sont les suivantes :

- « challenge » : contient la valeur A reçue de la part du serveur.
- « protocol » : contient le numéro du protocole utilisé par le client.

4- à la réception le serveur vérifie si le *Token* est bien celui qui a été envoyé. Si c'est le cas, il renvoie le message ascii « *Connected* » *A* où *A* est le nombre qui a été utilisé pour initialiser la connexion.

L'exemple suivant illustre une connexion réussie entre un client et un serveur après quelques tentatives d'envoi du message « GetToken », les messages préfixés « OUT » sont ceux envoyés, les messages préfixés « IN » sont ceux reçus.

**Client**

```
OUT     - GetToken 2901521149 Snake
OUT     - GetToken 2901521149 Snake
OUT     - GetToken 2901521149 Snake
OUT     - GetToken 2901521149 Snake
IN      - Token 494447609 2901521149 19
OUT     - Connect "\challenge\494447609\protocol\19"
IN      - Connected 494447609
```

**Serveur**

```
IN      - GetToken 2901521149 Snake
OUT     - Token 494447609 2901521149 19
IN      - Connect "\challenge\494447609\protocol\19"
OUT     - Connected 494447609
```

**Travail à faire**

Implémenter les deux phases décrite dans cette étape en python 2 et écrire une application cliente et une application serveur qui démontrent leur fonctionnement.