mickael.hoerdt@hesge.ch 2015/2016

Mini-projet d'ingénierie de protocoles réseaux informatiques

Un jeu de snake multi-joueur en réseaux

Étape 2: fiabilisation optionnelle du transport avec le protocole SnakePost

Certains messages du jeu du Snake multi-joueur devront être envoyés de manière fiable dans le canal de communication établi en étape 1. Par exemple, l'envoi de la position des pommes sur la surface de jeu du serveur aux clients, ou l'envoi du serveur aux clients de l'information d'un snake qui a perdu aux autres joueurs.

Pour cela, on se propose de concevoir un nouveau protocole appelé *SnakePost*. L'entête de ce protocole sera composé de 4 octets qui représentent deux entiers de 16 bits en binaire. Comme l'indique la figure 1, le premier entier dans l'entête représente un numéro de séquence, le deuxième un numéro d'accusé de réception.

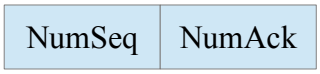

Figure 1 : Entête du protocole SnakePost

Comme l'indique la figure 2, le protocole SnakePost est destiné dans le jeu de Snake multiplayer à être transporté au dessus de *SnakeChan*, et en héritera donc de ses propriétés (**connecté**, **bidirectionnel** et **ordonnancé**) sans en être dépendant : il pourrait très bien fonctionner uniquement au dessus d'uniquement UDP par exemple, à condition bien entendu que les parties communicantes soient pré-configurées pour.

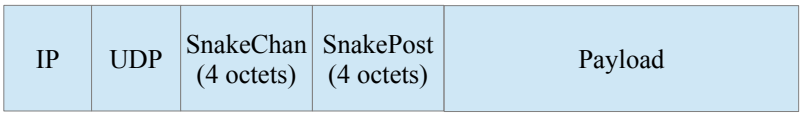

Figure 2 : Encapsulation des protocoles SnakeChan et SnakePost

SnakePost permettra, à la manière d'un service postal, d'envoyer certains messages avec garantie d'arrivée chez le destinataire et certains messages sans garantie d'arrivée chez le destinataire, selon la demande de service de l'application.

- Si l'application demande un transport non fiable, le numéro de séquence est mis à 0, ce qui indique au protocole que le message ne demande pas d'acquittement de la part du destinataire, dans ce cas là le traitement des messages est identique au protocole qui l'encapsule (SnakeChan, UDP ou autre).

- Si l'application demande un transport fiable le numéro de séquence sera initialisé à une valeur différente de 0. Ce numéro indique le numéro de séquence que le destinataire se chargera d'acquitter dès réception en copiant sa valeur dans le champs du numéro d'acquittement.

- SnakePost doit être capable de PiggyBacking[1], c'est à dire que si des données (fiables ou non) sont à envoyer au moment d'envoyer l'acquittement, SnakePost enverra les données et l'acquittement dans un même message.

- À tout moment, il ne peut y avoir qu'un seul message avec garantie d'arrivée en circulation dans un canal. Si l'application de SnakePost demande à envoyer un nouveau message fiable alors qu'il y en a déjà un en circulation, ce nouveau message sera stocké dans une file d'attente interne d'une taille de maximum 64 messages. Lorsque la file d'attente est pleine et qu'une application demande à envoyer un nouveau message SnakePost doit indiquer à l'application qui faudra réessayer plus tard, car la file d'attente du canal est pleine.

- Tant que le message n'aura pas été acquitté, celui-ci sera renvoyé par SnakePost à une fréquence relativement élevée, de l'ordre de 30 à 60 renvois par seconde, c'est à dire que la valeur du timer de nouvel essai sera situé entre entre 15 et 30 ms. Cette fréquence de renvoi est élevée afin de conserver la réactivité du jeu même lorsque un pourcentage élevé des messages est perdu par le réseau.

La figure 3 illustre le fonctionnement de SnakePost sur un diagramme temps séquence, avec Piggy-Backing :

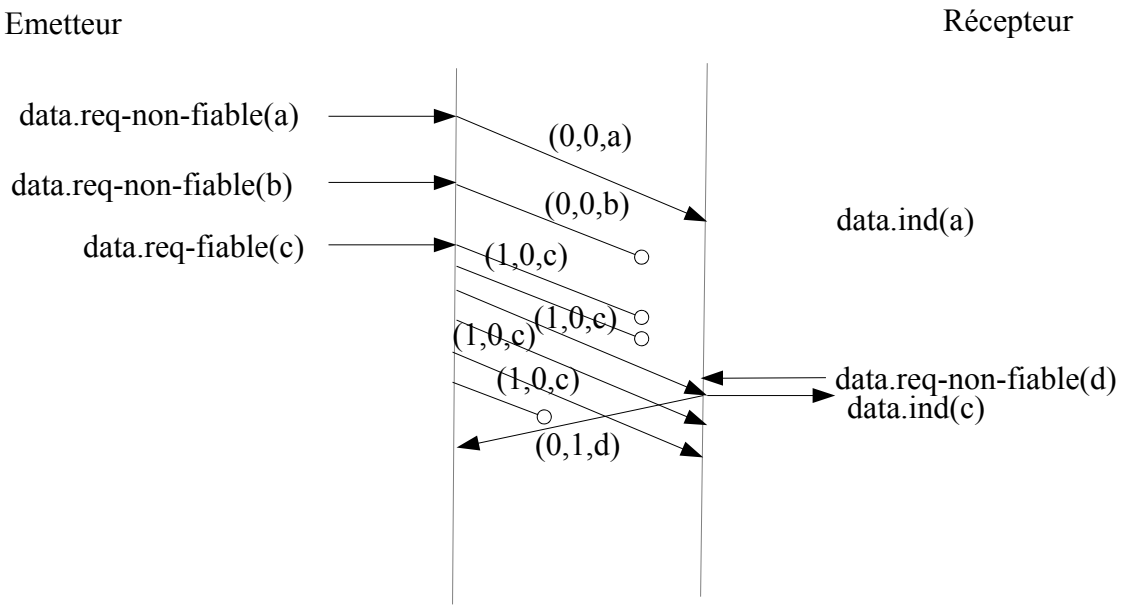

Figure 3 : Diagramme temps séquence du protocole SnakePost

Travail à faire

Implémenter le protocole SnakePost décrit dans cette étape en python 2 et écrire une application cliente et une application serveur qui démontrent son fonctionnement dans deux situations : la première avec SnakePost fonctionnant uniquement au dessus d'UDP, la deuxième avec SnakePost fonctionnant au dessus de SnakeChan.

1 https://en.wikipedia.org/wiki/Piggybacking_%28data_transmission%29