TP Applications Réparties
Java/NIO
RICM 4, 2017-2018, F. Boyer, P. Morat

1 – Baby-step
Complétez les classes fournies (client.java et server.java) pour mettre en oeuvre un ping-pong au dessus de nio. Votre client enverra un message « ping » et le serveur renverra systématiquement le message « pong », provoquant (après réception) le renvoi de « ping » par le client pour un échange infini.
Dans cette version préliminaire, on ne se souciera pas du fait qu'un interlocuteur (client ou serveur) puisse recevoir qu'une partie du message envoyé par l'autre interlocuteur.

2. Expérimentation plus avancée
Téléchargez et dé-zippez le paquet *NIO-DONNE-ETD.zip*. Ce paquet met en oeuvre un client/serveur nio plus élaboré que précédemment. Le serveur est associé à une interface graphique affichant des informations sur les échanges réseau. Un client graphique est également fourni.

2.1 - A partir des classes fournies, vous devez complétez le client et le serveur. En particulier, il vous faut tout d'abord compléter les classes ReadCont et WriteCont qui gèrent les continuations de traitement des événements de type read (resp. write). N'oubliez pas de programmer vos continuations comme des automates.

2.2 - Modifiez le code de la classe NioClient de manière à comptabiliser le nombre d'échanges (c.a.d le nombre de ping-pongs) pour chaque message. Vous utiliserez pour cela la méthode incrementExchange() de la classe Message.

2.3 - Classe ReadCont

a) Au niveau des continuations en lecture, complétez le code pour comptabiliser le nombre de pas qui ont été nécessaires pour recevoir chaque message.

b) Modifiez le code de la méthode handleRead() afin que celle-ci retourne un objet de type message contenant ce nombre de pas. Vous utiliserez le constructeur Message (bytes[]data, int nb_steps) pour cela.

c) En lançant plusieurs exécutions, essayez d'établir un lien entre la taille d'un message et le nombre de pas mis en oeuvre pour le recevoir complètement.