BPO
TP Application Client-Serveur - Partie 1

Sont fournis sur Arche :

- le diagramme de classes étudié en TD (au moins en partie ….) ;
- les classes **reseau.adresses.AdresseMac** et **reseau.adresses.Octet** ;
- les classes **reseau.adresses.Adresse** et **reseau.Message** (une version opérationnelle des classes développées dans le Tp précédent, pour ceux qui n'auraient pas fini ou auraient laissé des bugs) ;
- le squelette des autres classes de l'application ; certaines classes sont partiellement écrites ;
- la classe **reseau.tests.TestReseau** avec la fonction ***main***.

Mise en place de l'environnement de développement

Actuellement, vous avez un répertoire **reseau** contenant, entre autres, les classes **Adresse**, **Message** et **Octet**. Vous devez renommer ce répertoire, par exemple en **reseauAMoi.**
Ensuite, vous recopiez le nouveau répertoire **reseau** fourni sur Arche. Cela permet de débuter ce Tp sur des bases fiables en ce qui concerne **Adresse** et **Message**.

Il est impératif de suivre les questions suivantes dans l'ordre, sans chercher à vouloir remplir les classes à tout prix… Certaines questions sont l'occasion d'étudier du code fourni, d'autres sont l'occasion de compléter le code.

1) Lire le texte du constructeur de **reseau.tests.TestReseau** : il crée les différentes couches pour une machine donnée, et attache une application cliente à un port sur la couche transport. Compiler et exécuter cette classe. Que se passe-t-il lors de l'exécution ?

Pas de code à écrire … on complétera la classe plus tard.

2) Étudier la classe abstraite **reseau.couches.Couche** : elle contient des fonctions permettant de fixer les couches **plusUn** et **moinsUn** d'une couche. Compléter la fonction **toString**. En utilisant les fonctions **plusUn** et **moinsUn**, compléter le constructeur de **TestReseau** avec une séquence d'instructions qui relie les couches entre elles, pour la première machine seulement. S'assurer que la compilation se déroule sans problème.

3) Dans la classe **TestReseau**, écrire ensuite une séquence analogue qui crée une autre pile de protocoles pour une deuxième machine, avec une application serveur. Compléter par les instructions qui permettent de relier les deux couches **Ethernet** entre elles. S'assurer que la compilation se déroule sans problème.

4) Il n'est pas simple de tester directement le code écrit. Pour vérifier que la création des couches et des liens qui les

unissent s'est bien passée, on va envoyer un message. Dans le constructeur de **TestReseau**, la séquence qui permet d'envoyer ce message est déjà écrite (on l'a étudiée en TD). Exécuter la classe **TestReseau**. Que se passe-t-il ?

*Le message envoyé à **Transport** doit comporter 2 octets.*

> *Le message est envoyé par le client numérique à la couche **Transport** (avec la fonction **sendMessage**).*
> *Mais comme la fonction **sendMessage** n'est pas écrite, une exception est déclenchée. À noter l'utilisation indispensable du cast dans la fonction **sendMessage**, pour identifier clairement le type de **Couche** auquel on s'adresse, car les fonctions **sendMessage** n'ont pas toutes le même profil.*

5) Écrire le texte de la fonction **sendMessage** de la couche **Transport** : on ajoute un entête (appel de **getEntete**) avant de transmettre à la couche **Reseau**. Il faut donc aussi compléter la fonction **getEntete** de la classe **UDP**. Penser à afficher une trace de l'envoi, comme c'est fait dans la fonction **sendMessage** de **Application**. Vérifier le résultat de l'exécution de la classe **TestReseau**.

*Le message envoyé à **Reseau** doit comporter 10 octets.*

6) De la même façon, écrire le texte de la fonction **sendMessage** de la classe **IP** : on ajoute un entête avant de transmettre à la couche **Liaison**. On retrouve l'adresse Mac du destinataire en consultant la table **ARP**. Vérifier le résultat de l'exécution de la classe **TestReseau**.

*Le message envoyé à **Liaison** doit comporter 23 octets.*

7) Écrire enfin la fonction **sendMessage** de la classe **Ethernet** qui envoie une trame à son voisin, par l'intermédiaire de la fonction **receiveMessage**.Vérifier le résultat de l'exécution de la classe **TestReseau**.

*Le message envoyé à **Liaison** de son voisin doit comporter 35 octets.*

À ce stade, le message a été transmis de couche en couche sur la machine cliente. Reste à le faire remonter dans les couches de la machine serveur.

8) Écrire la fonction **receiveMessage** de la classe **Ethernet**. Cette fonction s'assure que le message est bien destiné à cette couche **Liaison**, en vérifiant l'adresse Mac du destinataire. Vérifier le résultat de l'exécution de la classe **TestReseau**.

9) Écrire la fonction **receiveMessage** de la classe **IP**. Vérifier le résultat de l'exécution de la classe **TestReseau**.

10) Écrire la fonction **receiveMessage** de la classe **Transport**. Pour retrouver la bonne application serveur à qui s'adresser, il faut consulter la table des applications, rangée dans **Transport**. Vérifier le résultat de l'exécution de la classe **TestReseau**.

> *On rappelle que l'association (multiplicité *) entre **Transport** et **Application**, permet à chaque couche transport de connaître toutes les applications via leur port.*

11) Écrire la fonction **receiveMessage** de la classe **Application** : elle traite le message (fonction **traiter**) et renvoie le résultat.

12) Modifier le code de la fonction **sendMessage** de la classe **IP**, de sorte que le message ne soit envoyé que si la machine destination fait partie du même réseau local que la machine source.

> *Rappel : il faut pour cela utiliser le masque : appliqué aux deux adresses IP (source et destination), il doit donner le même résultat.*

13) Ajouter un serveur **ServeurMaj** qui transforme tous les caractères d'une chaîne de caractères quelconque en lettres majuscules. Ce serveur sera installé sur un port (numéro à choisir) de la deuxième machine. Tester l'envoi d'un message (une chaîne de caractères) à ce nouveau serveur. *Remarque : votre chaîne de caractères ne comportera ni symbole ni caractères espace.*