École de technologie supérieure
Département de génie logiciel et des TI

Responsable du cours : **Nadjia Kara**
Chargés de cours : **Michel Houde, ing.**
Patrick Fernet, ing.
Chargés de laboratoire : **Maxime Champagne**
Steve Anicet Louokdom Takam
Session : **Hiver 2016**

GTI610 RÉSEAUX DE TÉLÉCOMMUNICATIONS

Laboratoire 4 : **Programmation des protocoles de communication à l'aide des sockets Java**

Durée = 9 h (3 séances)

Une démonstration du travail se fera pendant la troisième et dernière séance de laboratoire.

Objectif

Ce laboratoire se veut une introduction aux protocoles de la couche transport et de la couche application du modèle OSI. Ce laboratoire est divisé en deux parties.

La première partie vous permettra de vous familiariser avec l'établissement d'une connexion TCP et l'échange d'information entre deux processus applicatifs simples ainsi que de comprendre comment un serveur TCP associé à un port spécifique peut traiter plusieurs connexions à la fois.

La deuxième partie vous permettra de vous familiariser avec les protocoles UDP et DNS en vous demandant d'implémenter la partie communication manquante d'un serveur DNS.

Reportez-vous à la section « Rapport » pour des indications concernant la rédaction et la remise du rapport.

Première partie : client/serveur avec le protocole TCP

Pour réaliser cette étape, vous devez implémenter une application client et une application serveur. Les deux applications doivent communiquer entre elles et s'échanger de l'information à l'aide du protocole TCP. Vous devez utiliser les classes Socket et ServerSocket.

Le constructeur de socket permet de prendre en paramètre une adresse IP et un port. Lors de son instanciation, ce socket tentera automatiquement d'établir une connexion TCP à cette adresse et port.

Le constructeur ServerSocket permet de spécifier un port. Mais, après son instanciation, le socket ne sera pas en mode d'écoute pour autant. La méthode bloquante accept() fera en sorte que l'instance de ServerSocket sera prête à recevoir une connexion TCP.

La partie du serveur traitant le paquet reçu du client doit pouvoir s'exécuter dans un « thread » pour permettre l'exécution en parallèle de plusieurs instances de celui-ci.

Une fois la connexion établie entre un client et le serveur, l'application client doit attendre que des caractères soient entrés au clavier et, lorsqu'elle les reçoit, doit les envoyer au serveur. Ce dernier doit commencer par afficher le texte du message suivi de l'adresse IP du client et de son numéro de port. Ensuite, il doit réécrire le message reçu en majuscules puis le renvoyer au client. Ce dernier doit afficher la réponse du serveur sur la console.

Vous devez utiliser les méthodes getInputSteam() et getOutputStream pour recevoir et envoyer de l'information à travers les sockets. Ces méthodes manipulent les octets directement; rappelez-vous qu'il existe des classes tel que : InputStreamReader, BufferedReader et PrintWriter.

Question :

- Tout en joignant des bouts de code, expliquez comment vous avez réalisé cette partie. Vous devez expliquer ce que font le serveur et le client chacun de son côté et comment le multithread a été réalisé.

Deuxième partie : Serveur DNS

Cette partie permettra de vous familiariser avec le protocole de la couche transport UDP, ainsi qu'avec le protocole de la couche application DNS.

Un serveur DNS est un système hiérarchique distribué permettant de traduire des adresses symboliques en adresses IP. Pour cette partie, vous devez réaliser un serveur DNS simplifié qui répond aux requêtes de type standard (**opcode = 0000)** avec le type de donnée**s, type = 01** (adresse IP de l'hôte) et la classe de données **class = 01** (classe Internet). Pour vous aider dans cette réalisation, une application serveurDNS vous est fournie et on vous demande de compléter la méthode principale run() de la classe UDPReceiver. Cette dernière est en fait un « thread » qui reçoit les paquets UDP, analyse les messages DNS compris dans ces paquets et retourne une réponse au client où redirige la requête du client vers un autre serveur DNS.

Le protocole UDP fonctionne différemment du protocole TCP. Pour recevoir et envoyer de l'information, il faut utiliser les classes DatagramSocket et DatagramPacket. Les sockets UDP sont des DatagramSocket, et ils sont instanciés en spécifiant le port local auquel ils sont attachés. L'adresse et le port de destination sont spécifiés dans le DatagramPacket, à l'aide des méthodes setAdress() et setPort(). De plus, la méthode setData() permet de spécifier le contenu du DatagramPacket à envoyer.

La classe InetAddress permet de construire et de manipuler des adresses Internet.

Lors de son exécution, l'application prend comme argument l'adresse d'un autre serveur DNS. Ce dernier sera contacté si le nom de domaine ne peut pas être résolu localement. Vous pouvez choisir l'adresse du serveur DNS normalement utilisée par votre station et qui peut être obtenue en utilisant la commande « ipconfig /all », ou encore un des serveurs DNS publics; par exemple le serveur 8.8.8.8 de Google.

Parmi les classes fournies :

- La classe UDPAnswerPacketCreator permet de créer un paquet de réponse DNS grâce à la méthode CreateAnswerPacket. Cette méthode prend en paramètre un tableau d'octets et une liste de chaînes de caractères. Les chaînes de caractères représentent la ou les adresses IP associées au nom de domaine spécifié par le client. Pour le tableau d'octets, il s'agit du paquet de requête UDP reçu. Utiliser les objets ByteArrayInputStream et DataInputStream pour manipuler les octets du paquet, et la méthode getLength() pour savoir le nombre d'octets à lire.

- La classe QueryFinder permet de rechercher le nom de domaine dans votre fichier de correspondances (cache DNS).

- La classe AnswerRecorder, quant à elle, permet d'enregistrer une nouvelle correspondance nom de domaine-adresse(s)IP dans ce fichier.

La structure d'un message DNS est la suivante :

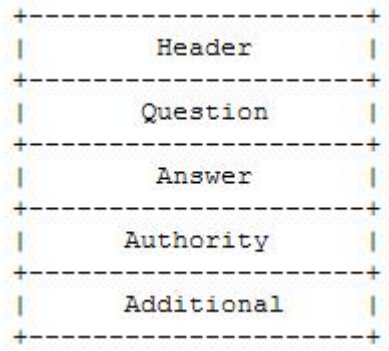

Figure 1 - Format d'un message DNS

Les champs qui sont importants dans le cadre de ce laboratoire sont « header », « question » et « answer ».

Le champ header est de la forme:

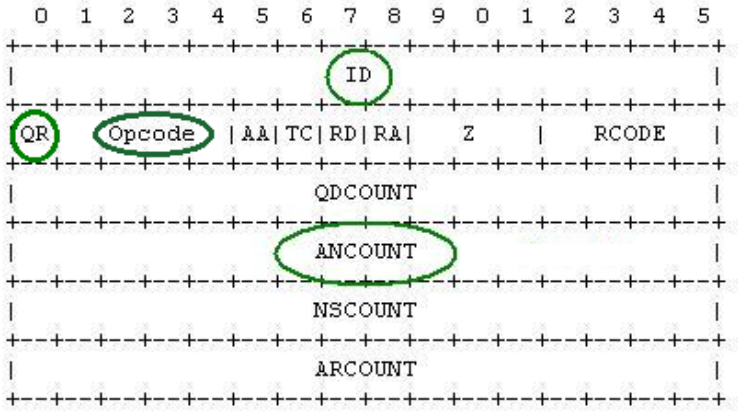

Figure 2 — Format du champ d'entête

Le champ ID (16 bits) permet de savoir de quelle requête il s'agit. Ce numéro est défini par le client lors de l'envoi de la requête, et ce numéro sera le même pour tous les messages DNS ayant rapport

avec la requête. Plus précisément, si une requête parvient à l'application serveur DNS et que celui-ci ne connaît pas l'adresse IP associée au nom de domaine demandé, il retransmet la requête vers un autre serveur DNS en utilisant le même ID. L'autre serveur DNS effectuera la résolution d'adresse et enverra une réponse DNS à l'application serveur DNS en utilisant le même ID. L'application serveur DNS fera parvenir cette réponse à la station ayant fait la requête en utilisant encore une fois le même ID.

Le champ QR (1 bit) permet d'indiquer s'il s'agit d'une requête (0) ou d'une réponse (1).

Le champ « ANCOUNT » spécifie le nombre de réponses qui suivra dans le message, c'est-à-dire le nombre d'entrées dans la section « Answer ».

À la suite de l'entête, on retrouve le champ question qui est de la forme :

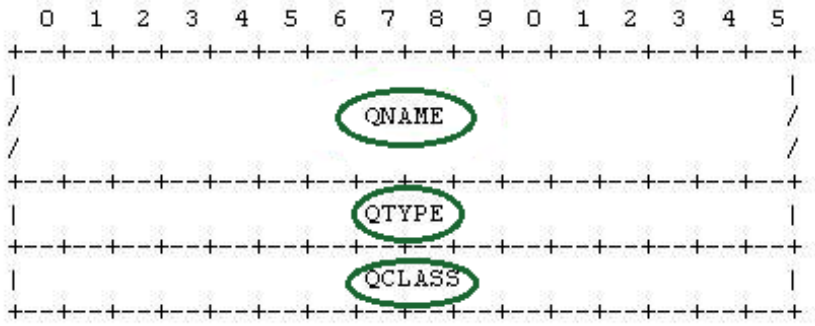

Figure 3 — Format du champ 'Question'

Le champ QNAME est le champ qui contient le nom de domaine pour lequel l'adresse IP correspondante est désirée. Le nom de domaine dans ce champ est inscrit section par section. Chacune des sections est précédée par un octet qui indique le nombre de caractères de cette section, donc le nombre d'octets lui étant réservés. Un octet contenant la valeur 0 termine le QNAME. Par exemple, le nom de domaine « etsmtl.ca » donne **06** 65 74 73 6d 74 6c **02** 63 61 **00**.

On peut lire des octets et les transformer en caractères à l'aide de CharactertoChars() en spécifiant un octet en paramètre. Il est important d'ajouter un point « . » entre chaque section du nom de domaine.

À la suite du champ question, on retrouve le champ « Answer » qui détient l'adresse IP associée au nom de domaine demandé. Voici son format :

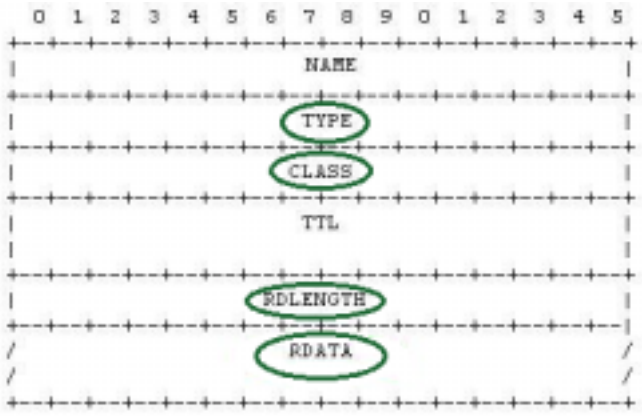

Figure 4 — Format du champ 'Réponse'

La section réponse peut avoir plusieurs formes. Rappelez-vous que pour ce laboratoire, on ne considère que les réponses de type A (type = 01) et de classe **IN (class = 01)**.

Le champ « RDLENGTH » donne la taille de l'adresse IP retournée. Donc, pensez à ne considérer que les adresses IPv4 dont le RDLENGTH = 4.

L'adresse IP recherchée est contenue dans la section RDATA. Une adresse IPv4 est sur quatre octets (chaque octet représente un nombre entre 0 et 255). Il faut donc pour la récupérer, ajouter un point « . » entre chacune des parties.

Pour plus de détails sur un message DNS, reportez-vous au fichier de présentation des sockets de communications disponible sur le site du cours.

Question :

- Tout en joignant des bouts de code, expliquez le fonctionnement de votre serveur DNS ainsi que les différents traitements à appliquer au paquet dans le cas d'une requête et dans le cas d'une réponse.

Pour tester votre DNS, vous pouvez utiliser les commandes :

nslookup

>server 127.0.0.1 qui redirige la requête DNS vers le serveur DNS s'exécutant sur la station locale.

>nomDeDomaine qui constitue le nom de domaine que vous voulez résoudre.

Démonstration

- Vous devez tester les deux parties du laboratoire.
- Pour la section serveur DNS, les trois types d'arguments seront testés :
  - default
  - [adresse DNS] <Fichier DNS> <TrueFalse/Redirection seulement>
  - showtable <Fichier DNS>
- **Aucun retard ne sera toléré.**

Rapport

- Répondez à toutes les questions tout en démontrant votre compréhension des sujets abordés.
- Le rapport doit contenir six sections :
  1. Une page couverture contenant au moins vos noms et le nom du laboratoire.
  2. Une introduction, qui pose le cadre du laboratoire.
  3. La réponse aux différentes questions du laboratoire en incluant l'énoncé de la question à laquelle vous répondez.
  4. Une discussion.
  5. Une conclusion.
  6. Les références aux livres, articles ou sites web utilisés.
- Il ne vous est pas demandé de refaire la théorie, mais de montrer, à travers vos réponses, que vous avez bien compris ce qui se passe dans les différents cas et pourquoi. N'oubliez pas que le rapport n'est pas jugé au poids, mais à la pertinence de vos réponses et analyses.

Remise du rapport

- Vous devez remettre une copie électronique de votre rapport sur le site Moodle du cours au plus tard une semaine après la fin du laboratoire.
- Le nom du fichier de remise doit être de la forme :
  GTI610_20161 labo4_Prénom1 Nom1_Prénom2 Nom2
- Vous devez également remettre une copie imprimée sur papier et le remettre dans la boite du département de génie logiciel et des TI (A-3085) prévue à cet effet.
- **Aucun retard ne sera toléré**. 20 % de la note sera retirée par jour de retard.
- Pour toute question, vous pouvez contacter votre chargé de laboratoire :
  - Gr. 01 : Maxime Champagne maxime.champagne94@gmail.com
  - Gr. 02 : Steve Anicet Louokdom Takam steve-anicet.louokdom-takam.1@ens.etsmtl.ca

Pondération

- Présentation et **qualité du français** : 5 %
- Introduction : 5 %
- Résultats, analyses et réponses aux questions : 35 %
- Démonstration : 40 %
- Discussion : 10 %
- Conclusion : 5 %