Prénom :
Nom :
Groupe TD :
Date :

Programmation répartie

M4102

François Merciol, Didier Lesage

DUT Informatique – 2nd année

Vannes – 2021/2022

Support de cours **étudiant**

http://m4102.merciol.fr/

(page gauche vide)

Table des matières

Table des figures

Liste des tableaux

Listings

1 Organisation du cours

1.1 Objectifs

Le module “Programmation répartie” (M4102) va nous permettre de nous entraîner et nous exercer à développer des programmes communiquant via un réseau. Cela signifie qu'il y aura des briques logicielles sur au moins deux sites distincts. Parfois ces sites seront hiérarchisés : un dominant / un dominé ou un serveur / un client. Parfois il n'y aura pas de hiérarchie : chacun jouant à son tour le rôle de producteur ou de consommateur d'information.

Il y aura principalement deux schémas de communication abordés :

- “socket” : communication que nous utiliseront principalement pour relier deux machines.
- “AJAX” : utilisé dans le cadre du web et permettant au navigateur de réaliser des traitements sur les données brutes fournies par le serveur.

Les développements se feront principalement en langage Java.

1.2 Calendrier

Les cours dureront sept semaines (en deux périodes de 6 et 1 semaines). Le rythme sera soutenu : 3 créneaux par semaines (4h30 par semaine, 31h30 au total). Les créneaux auraient dû se dérouler tous en salle machines mais en formation TD (groupe entier). Ils commenceront à distance dans un premier temps en espérant un retoure à l'IUT dès que possible. Il y aura une partie présentation des concepts puis compréhension par la pratique. Cette pratique alternant analyse et codage.

Nous aborderont :

- les “sockets”
  - TD1 : Clavardage, un exemple de serveur socket
  - TD2 : Mandataire, un exemple de serveur et de client socket
- le schéma “AJAX”
  - TD3 : Servlet, ou comment mettre en place “AJAX”
  - TD4 : Json, ou comment échanger des informations
  - TD5 : Jquery, une boîte à outil qui simplifie le développement en JavaScript
- client “HTTP” en Java
  - TD6 : Compréhension du DNS dynamique
  - TD7 : Mise à jour du DNS dynamique

1.3 Évaluation

Il y aura une évaluation continue :

- Les TD1 et TD2 seront évalués sur le travail réalisé en séance. Vous devrez donc rendre votre travail en fin de la 3e séance (donc vendredi). Pour ne pas charger la messagerie vous utiliserez :
  - soit les outils de l'université `https://filesender.renater.fr/`, `https://volumen.univ-ubs.fr/`
  - soit les outils développés en collaboration avec des étudiants de votre promotion en projet de synthèse `http://file.kaz.bzh/`

  Les copies seront corrigées par l'encvadrant du groupe A (même correcteur pour toute la promotion) durant le week-end pour que vous ayez vos notes la semaine suivante.

Il y aura une évaluation terminale :

- Elle portera sur un travail plus conséquent qui sera l'aboutissement de plusieurs TD. Il y aura 2 semaines de vacances entre le TD6 et TD7, ce qui pourra vous donner l'occasion de peaufiner votre travail. Cette évaluation sera réalisée par l'encadrant du groupe B (même correcteur pour toute la promotion).

Vous aurez donc 4 comptes-rendus à fournir :

- TD1 (Clavardage),
- TD2 (Mandataire),
- TD5 (Ajax),
- TD7 (DNS Dynamique).

1.4 Outils

Ne perdez pas de temps et économisez vos forces. Pour les comptes-rendus vous trouverez des modèles (les mêmes que pour le module M3101) sur `https://m4102.parlenet.org/supports/debut`. Commencez à remplir le document avant le TD.

Des modèles de programme sont disponibles également sur le site `https://m4102.parlenet.org/supports/debut`.

1.5 Conditions particulières

Ce module a été conçu pour la pratique et la mise en œuvre et le développement de composant communiquants entre les machines des binômes.

Il aurait dû se réaliser en salle TP. Le fait de devoir les réaliser à distances modifie considérablement l'apprentissage de ces notions.

Nous ferons de notre mieux pour l'adapter aux circonstances et faire preuve de bienveillance. Nous attendons également votre indulgence en cas de difficultés de communciation et d'équippement.

2 TD 1 : Clavardage

Rappels :

- N'oubliez pas qu'il y a un contrôle de présence. Les absences sont remontées et gérés par le secrétariat. Les retards seront comptabilisés pour être intégrés dans la note de contrôle continue.
- Après l'appel, c'est l'occasion de poser les questions concernant des éléments non compris du cours.
- N'oubliez pas qu'un compte-rendu sera demandé en fin de dernier créneau de la semaine (donc vendredi).
- Ce travail est à faire en binôme (un seul compte-rendu par binôme) dans le cas de groupe TD (environt 28).
- Si le travail est réalisé en TP (environt 14 étudiants) vous aurez
  - un bonus à rendre seul
  - un malus si des éléments sont copier-coller d'une autre copie.

  En d'autres termes si votre travail est collectif, assumez le en rendant en groupe, mais vous êtes invité à travailler seul pour mieux vous investir.
  Dans les circonstances actuelles de travail à domicile, il ne sera peut-être pas facile d'établir des binômes. Nous savons que vous ferez de votre mieux et ferons preuve de bienveillance en tenant compte des conditions.
- Vous ne devez pas rendre un simple listing, mais un document répondant aux questions posées (qui peut contenir des parties de listing). Nous vous invitons à joindre les sources à l'archive de votre compte-rendu.
- Vous rappellerez vos prénoms et noms, groupe TD, date et identifierez clairement le TP.
- Vous recopierez le texte des questions (et numéro) à chaque fois (pensez au copier-coller depuis l'énoncer).
- Vous répondrez par des phrases.
- Vous recopierez les commandes que vous avez saisies pour obtenir un résultat et plus généralement ce que vous saisissez dans un terminal.
- Quand c'est nécessaire, vous pouvez faire une copie d'écran (pas l'écran entier qui serait illisible, mais uniquement la partie que illustrant votre propos).
- Les modalités pour rendre votre travail pourront être précisées par votre encadrant (mel...). Dans tous les cas, vous devez privilégier un lien de téléchargement qui vous sera indiqué plutôt qu'une pièce-jointe à un courriel.

Ce TD porte sur la réalisation d'un programme faisant communiquer plusieurs clients et un serveur sur un réseau local (ou distant ☺). Pour ce faire, nous utiliserons des sockets au-dessus de TCP/IP. Le programme consistera en une communication instantanée simplifiée clavier/écran.

2.1 Présentation des sockets

Pour échanger des informations entre applications au-dessus d'IP, celles-ci utilisent des "prises" (en anglais *sockets*) en mode TCP ou bien UDP. C'est l'interface logicielle de communication. Dans ce TD, nous n'utiliserons que le mode TCP (i.e. le mode connecté).

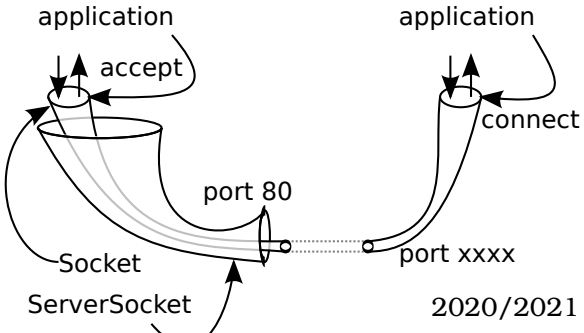
Un socket possède un identifiant sur 2 octets : le "port". Chaque port correspond à un usage d'application. L'usage se nomme service et la correspondance entre port et service se trouve dans le fichier `/etc/services`

Port inférieur à 1024

Les numéros inférieurs à 1024 ne sont utilisables que par l'administrateur. Cela garantit que le service n'est pas rendu par un utilisateur quelconque. Par exemple, il est préférable que les messages de tous les utilisateurs ne puissent pas être reçus par un invité sur une machine multi-utilisateur. C'est pourquoi le port 25 (*SMTP*) ne peut être ouvert que par une personne habilité du domaine concerné.

Lorsque l'utilisateur n'a pas de préférence de numéro, il utilise la valeur 0, un port disponible (au-dessus de 1024) lui est fourni.

L'application serveur web (par exemple apache) a un port (en général 80 (*HTTP*)). L'application navigateur (par exemple firefox) en a un également (il changera à chaque connexion). En mode connecté (TCP), l'application crée un socket particulier : le serveur de sockets (une sorte de "standard" d'appel).



Au moment de la connexion, le “standard” va créer une nouvelle liaison pour pouvoir libérer la ligne. Elle aura le même numéro de port côté serveur. Les liaisons sont différenciées par le fait qu'elles associent un numéro de port client et un numéro de port serveur.

En Java, la classe *ServeurSocket* représente le “standard” d'appel.
A la construction le numéro de port est fourni (l'adresse IP est trouvée automatiquement localement). Si vous n'êtes pas connecté à un réseau ce sera *localhost* (127.0.0.1) Si vous êtes connecté, ce sera l'adresse de la carte réseau de connexion.

La méthode *accept* est bloquante. Elle sera libérée par une connexion d'un client. Elle retourne un *socket* déjà connecté.

Les entrées/sorties en mode flux sont retrouvées à l'aide des méthodes : “getInputStream” et “getOutputStream”.

Voici un exemple de code qui crée un serveur de socket (téléchargeable sur le site `http://m4102.parlenet.org/codes/`).

```
      int port = Integer.parseInt (args [0]);
      ServerSocket serverSocket = new ServerSocket (port);
      System.err.println ("Start server on port "+port);

      for (;;) {
    Socket call = serverSocket.accept ();
    System.err.println ("Client comming from "+
            call.getRemoteSocketAddress ());
    BufferedReader in =
        new BufferedReader
        (new InputStreamReader (call.getInputStream ()));
    PrintStream out =
        new PrintStream (call.getOutputStream (), true);
    BufferedReader sysin =
        new BufferedReader (new InputStreamReader (System.in));
    for (;;) {
        String line1 = in.readLine ();
        if (line1 == null)
      break;
        System.out.println (line1);
        System.out.println ();
        String line2 = sysin.readLine ();
        out.println (line2);
        out.println ();
    }
    System.err.println ("Socket closed");
      }
```

Listing 1 – Code Java d'un serveur de sockets

Les protocoles de l'Internet sont à l'origine sous forme d'échanges de commandes texte ridumentaire (en anglais). Voici un exemple d'échange entre un navigateur et un serveur web :

```
GET /index.html HTTP/1.1¶
Host: m4102.parlenet.org¶
Accept-Language: argoTerrien¶
User-Agent: Zorglub¶
¶
```

```
HTTP/1.1 200 OK¶
Date: Sun, 5 Oct 2003 11:00:07¶
Server: Apache/2.0.40¶
Accept-Ranges: bytes¶
Content-Length: 2898¶
Connection: close¶
Content-Type: text/html¶
¶
<!DOCTYPE HTML PUBLIC...
```

Notons que la structure est la même pour le client et le serveur :

- une ligne de protocole composée de 3 mots
- un en-tête MIME (Multipurpose Internet Mail Extensions). Il s'agit d'une suite de lignes associant variable et valeur. L'en-tête se termine par une ligne vide.
- les données échangées :
  - valeurs de formulaire du client vers le serveur ou
  - page demandée du serveur vers le client.

La ligne de protocole est différente entre le client et le serveur. Pour le client, c'est un nom de méthode (GET, PUT, POST...) suivi d'une URI (chemin local d'informations) et terminé par le protocole utilisé. Pour le serveur, c'est le protocole suivi d'un code d'erreur et terminé d'une traduction abrégée en anglais.

URL ou URI ?

- L'URI est le chemin local de l'information sur le serveur (qui peut être différent de l'arboressence de fichier)
- l'URL est le chemin unique sur l'Inernet (protocole, adresse IP, port, URI, QUERY_STRING...)

Cas de l'hébergement multiple

Notez bien que le protocole HTTP/1.0 permet de préciser dans l'en-tête le nom du serveur final qui fournit les pages (dans le cas de serveurs mutualisés). Par exemple :

```
Host: m4102.parlenet.org
```

C'est ce qui permet à un même serveur physique d'héberger plusieurs serveurs web.

Cas d'un mandataire

Notez également que dans le cas de mandataire (en anglais *proxy*), le chemin n'est pas un chemin local (URI) mais le lien complet avec le nom du serveur (URL). Par exemple :

```
GET http://m4102.parlenet.org/ HTTP/1.0
```

C'est ce qui permet au mandataire de savoir à quel serveur physique il doit s'adresser.

Il existe un outil efficace qui connecte un socket à un écran et un clavier : `telnet` (aujourd'hui remplacé par `nc` avec les même paramètres). Par exemple :

```
telnet localost 4141
```

2.2 Compréhension des sockets

Question : Donnez la suite d'instructions pour rechercher la page `http://m4102.parlenet.org/debut` sans proxy

Réponse : ✍

Question : Donnez la suite d'instructions pour rechercher la page `http://m4102.parlenet.org/debut` avec le proxy squidva.univ-ubs.fr :3128

Réponse : ✍

2.3 Compréhension du sujet

Le logiciel résultant de ce TD possède les propriétés suivantes :

- Le programme est écrit en Java.
- Le programme ouvre un port en mode serveur.
- Le serveur accepte un nombre indéfini d'utilisateurs.
- Lorsqu'un utilisateur se connecte le serveur lui demande son nom, puis attend un nombre indéfini de messages (lignes).
- Chaque message envoyé par un utilisateur est retransmis à tous les autres, précédé par le nom de l'utilisateur.
- Les utilisateurs se connectent en utilisant simplement la commande « telnet » (ou « nc »).

2.4 Démonstration et exercice

Vous pouvez vous rendre mieux compte des spécifications en faisant un essai.

- Ouvrez un terminal
- Saisissez la commande :
  - `nc clavardage.parlenet.org 4141` (pour le groupe A le port 4141)
  - `nc clavardage.parlenet.org 4242` (pour le groupe B le port 4242)
- Vous recevez une invitation “What’s your name ?”
- Saisissez **votre groupe TD** et **votre nom** : `GrpA X.Durand` (ces identifiants seront utilisés dans les traces du serveur pour vérifier que vous avez réussi l’exercice)
- Vous aurez en retoure un chaleureu message de bienvenu
- Vous pouvez ensuite saisir du texte qui sera répercuté vers tous les autres utilisateurs connectés.
- Il existe deux commandes particulières :
  - pour connaître la liste des autres participant (mot unique sur la ligne) : `LIST`
  - pour quitter la session (mot unique sur la ligne) : `QUIT`

2.5 Analyse du logiciel

Question : Dire combien de tâches devront être mises en œuvre et quels seront leurs rôles.

Réponse : ✍

Question : Décrire en pseudo-code le comportement des différentes tâches.

Réponse : ✍

Question : Quelles sont les informations gérées par le serveur ?

Réponse : ✍

Question : Décrire les risques de conflit qui peuvent apparaître.

Réponse : ✍

2.6 Conception du logiciel

Nous abordons la phase de réalisation.

Question : Combien faut-il écrire de programmes (ainsi que ce qui peut être réutilisé d’Unix) ?

Réponse : ✍

Question : Y a-t-il des activités en concurrence pendant l’utilisation de la messagerie instantanée par 3 utilisateurs ?

Réponse : ✍

Question : Comment le serveur accepte-t-il les connexions des utilisateurs ?

Réponse : ✍

Question : Lorsqu’un utilisateur envoie un texte, que doit faire le serveur ?

Réponse : ✍

Question : Quelles sont les informations (ou données) que l'on doit mémoriser par utilisateur et qui sont nécessaires à sa création (âge, nom, profession, canal de lecture...)? Ils constituront la base de données manipulée par le serveur.

Réponse : ✍

Question : Écrire la classe "User" qui représente un utilisateur chez le serveur (attribut et constructeur). Les exemplaires de cette classe constituent la base de données du serveur.

Réponse : ✍

Question : Comment le serveur peut-il gérer la diffusion d'un texte envoyé par un utilisateur (attribut et méthode)? Dans la classe Clavardage

Réponse : ✍

Question : Où placer l'activité d'attente de texte envoyé par un utilisateur (écrire la méthode)?

Réponse : ✍

Question : Modifier le constructeur de la classe Utilisateur pour ajouter automatiquement un utilisateur dans la base de données du serveur et lancer son activité.

Réponse : ✍

Question : Écrire le constructeur du serveur et l'attente de nouveaux utilisateurs.

Réponse : ✍

Question : Écrire la gestion de l'ensemble des utilisateurs dans le serveur. Comment a-t-on réglé le problème d'accès concurrence à la base des utilisateurs?

Réponse : ✍

Question : Faire un lanceur qui prenne en argument le port d'écoute.

Réponse : ✍

3 TP1 : Clavardage

Cette section porte sur le compte-rendu que vous avez à fournir. Repportez-vous aux rappels du début de la section du TD.

Que dois-je faire ?

Vous retrouverez une partie des questions du TD, mais pas nécessairement dans le même ordre. Il est donc important dans votre compte-rendu de recopier les questions. Ce sont ces questions et ces réponses que l'on attend de vous.

Format des archives

Le nom de l'archive doit être informatif (année, module, nom du TP, groupe TD, tous les noms des auteurs) comme :

`2021-M4102-Clavardage-GrpA-X.Dupond-Y.Durand.tar.bz2`

L'archive doit contenir un répertoire avec le même format de nom :

`2021-M4102-Clavardage-GrpA-X.Dupond-Y.Durand.`

L'encadrant doit pouvoir extraire toutes les archives dans le même espace sans que les fichiers ne se mélangent.

3.1 Exercices de compréhension

Voici quelques exercices pour la prise en main de telnet. Donnez les commandes en shell et décrivez le résultat.

Il s'agit des mêmes questions qu'en TD. Mais cette fois vous devez donner des traces de vos exécutions (ou des explications en cas d'erreur).

Question : Donnez la suite d'instructions pour rechercher la page `http://m4102.parlenet.org/debut` sans proxy

Réponse : ✍

Question : Donnez la suite d'instructions pour rechercher la page `http://m4102.parlenet.org/debut` avec le proxy squidva.univ-ubs.fr :3128

Réponse : ✍

3.2 Validation des outils

IPV4 avec Java

Vous pouvez rencontrer des difficultés de connexions avec un système gérant en même temps IPV6 et IPV4. Dans ce cas utilisez l'option : `java -Djava.net.preferIPv4Stack=true -cp ....`

Question : Téléchargez le programme `recvcmd.java` sur le site `http://m4102.parlenet.org` et testez-le avec le port 8080. Comment vous connectez-vous à lui ?

Réponse : ✍

3.3 Compréhension du logiciel

Question : Dire combien de tâches devront être mises en œuvre et quels seront leurs rôles.

Réponse : ✍

Question : Décrire en pseudo-code le comportement des différentes tâches.

Réponse : ✍

Question : Décrire les risques de conflit qui peuvent apparaître.

Réponse : ✍

Question : Combien faut-il écrire de programmes (ainsi que ce qui peut être réutilisé d'Unix) ?

Réponse : ✍

3.4 Réalisation du logiciel

Consultez l'API Java concernant les classes : String, Integer, Vector<E>, ServerSocket, Socket, IOException, Thread, PrintStream, BufferedReader, InputStreamReader

Pour réaliser cet exercice, j'ai utilisé les mots clefs Java : break, catch, class, continue, extends, for, if, import, int, new, package, private, public, static, synchronized, this, true, try, void

Réalisez le programme par étape en insérant votre code en dessous des questions suivantes :

Question : Faire un lanceur qui prenne en argument le port d'écoute.

Réponse : ✍

Question : Écrire le constructeur du serveur et l'attente de nouveaux utilisateurs.

Réponse : ✍

Question : Écrire la gestion de l'ensemble des utilisateurs dans le serveur. Comment a-t-on réglé le problème d'accès concurrence à la base des utilisateurs ?

Réponse : ✍

Question : Écrire la classe "User" qui représente un utilisateur chez le serveur (attribut et constructeur). Les exemplaires de cette classe constituent la base de données du serveur.

Réponse : ✍

Question : Ecrire l'activité d'attente de texte envoyé par un utilisateur (écrire la méthode) ?

Réponse : ✍

3.5 Test du logiciel

Question : Donnez des exemples de fonctionnement.

Réponse : ✍

Question : Quelles sont les limites ? Quelles sont les améliorations à apporter ?

Réponse : ✍

N'oublez pas de joindre les sources et de respecter les formats de nom de l'archive et du répertoire qu'il contient.

4 TD 2 : Mandataire

Rappels :

- N'oubliez pas qu'il y a un contrôle de présence. Les absences sont remontées et gérés par le secrétariat. Les retards seront comptabilisés pour être intégrés dans la note de contrôle continue.
- Après l'appel, c'est l'occasion de poser les questions concernant des éléments non compris du cours.
- N'oubliez pas qu'un compte-rendu sera demandé en fin de dernier créneau de la semaine (donc vendredi).
- Ce travail est à faire en binôme (un seul compte-rendu par binôme) dans le cas de groupe TD (environt 28).
- Si le travail est réalisé en TP (environt 14 étudiants) vous aurez
  - un bonus à rendre seul
  - un malus si des éléments sont copier-coller d'une autre copie.

  En d'autres termes si votre travail est collectif, assumez le en rendant en groupe, mais vous êtes invité à travailler seul pour mieux vous investir.
  Dans les circonstances actuelles de travail à domicile, il ne sera peut-être pas facile d'établir des binômes. Nous savons que vous ferez de votre mieux et ferons preuve de bienveillance en tenant compte des conditions.
- Vous ne devez pas rendre un simple listing, mais un document répondant aux questions posées (qui peut contenir des parties de listing). Nous vous invitons à joindre les sources à l'archive de votre compte-rendu.
- Vous rappellerez vos prénoms et noms, groupe TD, date et identifierez clairement le TP.
- Vous recopierez le texte des questions (et numéro) à chaque fois (pensez au copier-coller depuis l'énoncer).
- Vous répondrez par des phrases.
- Vous recopierez les commandes que vous avez saisies pour obtenir un résultat et plus généralement ce que vous saisissez dans un terminal.
- Quand c'est nécessaire, vous pouvez faire une copie d'écran (pas l'écran entier qui serait illisible, mais uniquement la partie que illustrant votre propos).
- Les modalités pour rendre votre travail pourront être précisées par votre encadrant (mel...). Dans tous les cas, vous devez privilégier un lien de téléchargement qui vous sera indiqué plutôt qu'une pièce-jointe à un courriel.

Ce TD porte sur la réalisation d'un programme faisant communiquer des clients et des serveurs sur un réseau. Pour ce faire, nous utiliserons des sockets au-dessus d'IP. Le programme consistera en une communication permettant de tracer les échanges entre un client et un serveur.

De plus, il vous servira de boîte à outils pour l'analyse de protocole que vous aurez à mettre au point dans l'avenir.

4.1 Complément sur les sockets

Commençons par un complément de cours. La semaine dernière nous avons vu la mise en place d'un serveur de sockets en Java. Cette semaine nous examinons la connexion depuis un programme Java vers un serveur déjà existant. Voici un exemple de code qui crée un client de socket.

```
    Socket clientSocket =
  new Socket (args [0], Integer.parseInt (args [1]));
    PrintStream out =
  new PrintStream (clientSocket.getOutputStream (), true);
    out.println (args[2]);
    out.flush ();
    clientSocket.shutdownOutput ();
    BufferedReader in =
  new BufferedReader
  (new InputStreamReader (clientSocket.getInputStream ()));
    for (;;) {
  String line = in.readLine ();
  if (line == null)
      break;
  System.out.println (line);
  System.out.flush ();
  out.close ();
    }
```

Listing 2 – Code Java d'un client de socket

Comme nous connaissons maintenant les deux facettes de la communication avec sockets (client et serveur), nous allons réaliser un programmme qui met en œuvre les deux aspects synultanément.

4.2 Compréhension du sujet

Le logiciel résultant de ce TD possède les propriétés suivantes :

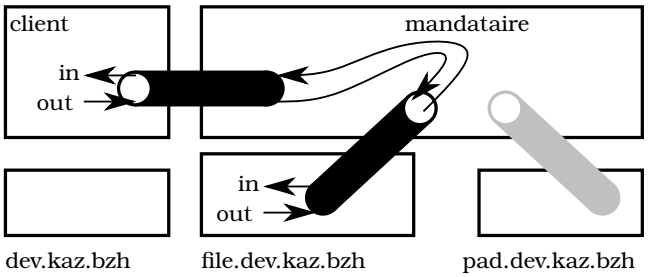
- un comportement de serveur avec un port de connexion paramétrable,
- un comportement de client vers d'autres services (machine et port paramétrables),
- une retransmission "full duplex", c'est-à-dire simultanée et dans les deux sens de transmission entre le client et le serveur,
- une indépendance du protocole retransmis,
- l'acceptation de plusieurs connexions simultanées,
- la possibilité de traces (montante ou descendante)
- la possibilité de modification à la volée (dans le sens descendant) d'une chaîne de caractères par une autre

Nous ajoutons une caractéristique supplémentaire :

- Il est possible de considérer la première ligne et de l'annalyser pour choisir le serveur que l'on va relier.

Ceci nous permet de réaliser un mandataire web ("Proxy" en anglais).

Le schéma général est le suivant :



Le client est par exemple un navigateur "firefox". Vous le configurez pour qu'il passe par un mandataire (proxy) comme indiqué dans l'énoncé du TP.

Lorsque vous chargez une page comme "dev.kaz.bzh" :

- Votre navigateur va se connecter avec un socket vers votre mandataire.
- Ce dernier lit le début du flut pour extraire la première ligne :

  `GET http://dev.kaz.bzh/xxx HTTP/1.1`
- Il va extraire le serveur visé "dev.kaz.bzh".
- Il va ouvrir à son tour un socket vers ce serveur et recopier la première ligne altérée de la façon suivante :

  `GET /xxx HTTP/1.1`
- Il va ensuite recopier les données qui provienne du client vers le serveur et les données qui viennent du serveur vers le client.

Plusieurs remarques :

- La copie des données du client vers le serveur ne doivent pas être bloquantes pour le reste du programme (donc une tâche (Thread) est nécessaire)
- La copie des données du serveur vers le client ne doivent pas être bloquantes pour le reste du programme (donc une tâche (Thread) est nécessaire)
- Les données copiées sont brutes et de n'importe quels formats : images, sons... (non textuelle ou binaire si vous préférez). Il ne sera pas possible d'utiliser les classes de la semaine dernière, mais les class "DataInputStream" et "DataOutputStream".
- À une entrée du client correspond une sortie vers un serveur.
- À une sortie du client correspond une entrée du serveur.
- Le mécanisme de copie est semblable dans un sens et dans l'autre (mutualisez le code).
- Si une source se tarit, elle doit fermer le puit (le consommateur) correspondant (l'autre socket)

Liaison non chiffrée

Les exemples que nous allons montrer sont en “HTTP” et non “HTTPS”. Il est important que les échanges soient en clair pour que nous puissions analyser le contenu. C’est pour cela que nous vous conseillons d’utiliser le site “dev.kaz.bzh”.

4.3 Analyse du logiciel

Question : Donnez un schéma d’utilisation d’un mandataire (par exemple web). Décrivez la communication entre le navigateur et le serveur au travers du mandataire. Expliquez comment notre mandataire vient s’insérer.

Réponse : ✍

Question : Comment paramétrer le mandataire ?

Réponse : ✍

Question : Pour une requête combien y a-t-il de canaux et de sockets chez le mandataire ?

Réponse : ✍

Question : Dans quel ordre les informations arrivent-elles sur les canaux ? Connaît-on leurs tailles ? Que doit faire le mandataire avec ces données ? Y a-t-il un ordre séquentiel de traitement ?

Réponse : ✍

Question : Écrivez une méthode “read” qui alloue et rempli un tampon mémoire avec l’entrée et une méthode “write” qui copie ce tempon dans la sortie. Déclarez les attributs nécessaires. Pour information :

- La méthode in.read (b) retourne le nombre d’octets lus dans un tableau
- La méthode out.write (b, 0, n) écrit les n premiers octets d’un tableau

---

Réponse : ✍

Question : Écrire une méthode qui copy le flux d’entrée sur celui de sortie en utilisant les méthdoe précédentes.

Réponse : ✍

Question : Donnez l’expression régulière permettant l’analyse de la ligne HTTP du client.

Indiquez également comment elle est écrite dans le fichier source Java.

Réponse : ✍

Question : Que faire lors d'une demande de requête d'un client ? (fournir le code correspondant)

- On utilisera les classes DataOutputStream, DataInputStream

Réponse : ✍

Question : Ecrivez une méthode qui met à jour le tampon mémoire déjà lu.

Réponse : ✍

Question : Que faire lors d'une réponse du serveur ? (fournir le code correspondant)

Réponse : ✍

Question : Que faut-il faire pour disposer d'une trace des requêtes émises ?

Réponse : ✍

Question : Que faut-il faire pour réaliser de la désinformation en ligne ?

Réponse : ✍

Question : Que faire pour en faire un outil d'espionnage de mot de passe de messagerie.

Réponse : ✍

5 TP2 : Mandataire

Cette section porte sur le compte-rendu que vous avez à fournir. Repportez-vous aux rappels du début de la section du TD.

Que dois-je faire ?

Vous retrouverez une partie des questions du TD, mais pas nécessairement dans le même ordre. Il est donc important dans votre compte-rendu de recopier les questions. Ce sont ces questions et ces réponses que l'on attend de vous.

Format des archives

Le nom de l'archive doit être informatif (année, module, nom du TP, groupe TD, tous les noms des auteurs) comme :

`2021-M4102-Clavardage-GrpA-X.Dupond-Y.Durand.tar.bz2`

L'archive doit contenir un répertoire avec le même format de nom :

`2021-M4102-Clavardage-GrpA-X.Dupond-Y.Durand.`

L'encadrant doit pouvoir extraire toutes les archives dans le même espace sans que les fichiers ne se mélangent.

5.1 Validation des outils

Question : Reprenez le programme `recvcmd.java` de la dernière séance avec le port 8080. Téléchargez le programme `sendcmd.java` et testez-le avec le serveur précédent.

Réponse : ✍

5.2 Compréhension du sujet

Question : Donnez un schéma d'utilisation d'un mandataire (par exemple web). Décrivez la communication entre le navigateur et le serveur au travers du mandataire. Expliquez comment notre mandataire vient s'insérer.

Réponse : ✍

Question : Pour une requête combien y a-t-il de canaux et de sockets chez le mandataire ?

Réponse : ✍

5.3 Réalisation du logiciel

Consultez l'API Java concernant les classes : String, Integer, ServerSocket, Socket, IOException, DataInputStream, DataOutputStream, Thread

Question : Faire un lanceur qui prend les paramètres nécessaires en arguments.

Réponse : ✍

Question : Ecrire le ou les contsructeurs de la classe du Mandataire et l'activité d'attente des clients.

Réponse : ✍

Question : Écrivez une méthode “read” qui alloue et rempli un tampon mémoire avec l’entrée et une méthode “write” qui copie ce tempon dans la sortie. Déclarez les attributs nécessaires. Pour information :

- La méthode in.read (b) retourne le nombre d’octets lus dans un tableau
- La méthode out.write (b, 0, n) écrit les n premiers octets d’un tableau

---

Réponse : ✍

Question : Écrire une méthode qui copy le flux d’entrée sur celui de sortie en utilisant les méthdoe précédentes.

Réponse : ✍

Question : Donnez l’expression régulière permettant l’analyse de la ligne HTTP du client.

Indiquez également comment elle est écrite dans le fichier source Java.

Réponse : ✍

Question : Que faire lors d’une demande de requête d’un client ? (fournir le code correspondant)

- On utilisera les classes DataOutputStream, DataInputStream

Réponse : ✍

Question : Ecrivez une méthode qui met à jour le tampon mémoire déjà lu.

Réponse : ✍

Question : Que faire lors d’une réponse du serveur ? (fournir le code correspondant)

Réponse : ✍

Question : Que faut-il faire pour disposer d’une trace des requêtes émises ?

Réponse : ✍

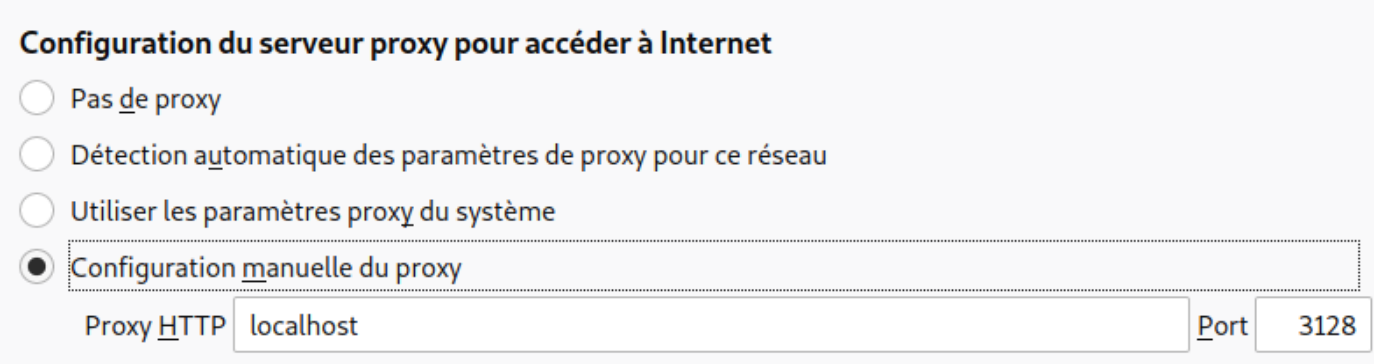
Question : Que faut-il faire pour réaliser de la désinformation en ligne ?

Réponse : ✍

5.4 Test du logiciel

Configuration du navigateur

Pour le test il faudra configurer votre navigateur pour utiliser votre mandataire. Pour ce faire allez dans le menu de votre navigateur firefox : “Edition” => “Préférences”. Puis dans la page “about :preferences” choisissez l’onglet “Général”. En bas de page, dans la section “Paramètres réseau”, cliquez sur le bouton “Paramètres...”. Choisissez la “Configuration manuelle du proxy” pour renseigner les informations “localhost” et “3128”. Puis validez. Vous devez obtenir un formulaire comme suit :



Question : Donnez des exemples de fonctionnement.

- lancez votre mandataire sur le port 3128
- configurez votre navigateur
- allez sur http ://dev.kaz.bzh :80/

Recherchez le mot “KAZ”.

Réponse : ✍

Configuration du navigateur

Pensez à réinitialiser la configuration avec “Utiliser les paramètres proxy du système” de votre navigateur. Sinon, dès que votre programme Java sera arrêté vous ne pourrez plus naviguer.

Question : Quelles sont les limites ? Quelles sont les améliorations à apporter à votre logiciel de “désinformation” en ligne ?

Réponse : ✍

Question : Que faire pour en faire un outil d’espionnage de mot de passe de messagerie.

Réponse : ✍