# TP 5 HTTP

Ilyas Bambrik

# Table des matières

# Travail demandé

*Pour chaque capture d'écran existante dans cette fiche TP, vous devez avoir la même capture afin de répondre aux questions lors de la consultation.*

# Demos HTTP

II



## 1. HTTPs

*Remarque*

Avant de commencer le TP, télécharger les programmes utilisés depuis le lien suivant. Le répertoire python 2 contient les programmes inscrits dans cette fiche et le répertoire python 3 contient les programmes équivalents en python 3 (utilisez la version qui vous convient).

Programme Python 2 et Python 3 (téléchargez l'un des deux répertoires selon votre version):

*https://drive.google.com/open?id=1v6TINpuE8PKvUCtK6KJsQwSvhFHw4y7-*

- Les démonstrations suivantes montrent comment accéder à des ressources HTTP. Cependant, ces requêtes ne s'exécuteront pas pour un site HTTPS (la communication HTTPS nécessite deux autres étapes supplémentaires qui vont être expliqués prochainement) ;
- La librairie urllib permet d'exécuter les requêtes http d'une manière plus simple, mais pour comprendre comment ça fonctionne au bas niveau, ces démonstrations utilisent les sockets ;

## 2. Requête GET

Le code suivant connecte à un serveur HTTP (*www.tcpipguide.com*) et envoie une simple requête GET (si vous essayez d'accéder à un autre serveur web, utilisez le nom du serveur web correspondant et remplacez *www.tcpipguide.com* dans le code et la requête HTTP). La requête dans cet exemple demande la *page d'accueil* de ce serveur web (/):

```python
# Code Python 2
# -*- @author: Ilyas Bambrik -*-

import socket
ConnexionAuServeurHTTP=socket.socket()

# nom de domaine du serveur web

Serveur="www.tcpipguide.com"
# numero de port
Port=80

# connecte au serveur distant
ConnexionAuServeurHTTP.connect((Serveur,Port))
# transmet la requete GET. Les headers de la requetes se terminent pas CRLF (\r\n)
# la requete se termine par CRLFCRLF (\r\n\r\n)
ConnexionAuServeurHTTP.send("GET / HTTP/1.1\r\nHost:www.tcpipguide.com\r\nConnection:close\r\n\r\n")
# Les deux entetes HTTP dans cette requete sont :
# 1) [Host:www.tcpipguide.com] (nomdu domaine du serveur web)
# 2) [Connexion:close] (pour cloturer la connection apres l'envoie de la reponse)

# recevoir le contenu de la page ( lire le contenu reçu du buffer en morceaux /sequence d'octets [chaine de caractere] de taille maximale <= 1024 )
rec=ConnexionAuServeurHTTP.recv(1024)
while True:
    # apres la cloture de la connexion, le nombre d'octets lus sera d'une taille de 0
    if len(rec)==0:
        break # sortire de la boucle si la taille du message reçu == 0
    # imprime le contenu du morceau reçu
    print rec,
    # lire un nouveau morceau
    rec=ConnexionAuServeurHTTP.recv(1024)

```

Le résultat de la requête sera la réponse suivante (entêtes de la réponse serveur web + la page web renvoyée par le serveur) :


Python 2.7.9 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.9 (default, Dec 10 2014, 12:24:55) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
HTTP/1.1 200 OK
Date: Thu, 13 Dec 2018 18:39:20 GMT
Server: Apache/2.4.37
Accept-Ranges: bytes
Content-Length: 20757
Content-Type: text/html
Réponse Serveur avec code 200
et taille de la ressource
CRLFCRLF pour marquer le début de la ressource transmise par le serveur
<html>
<head>
<meta http-equiv="Content-Language" content="en-us">
<meta name="GENERATOR" content="Microsoft FrontPage 6.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Welcome to The TCP/IP Guide!</title>
<script language="JavaScript" fptype="dynamicanimation">
<!--
function dynAnimation() {}
function clickSwapImg() {}
//-->

# 3. Requête HEAD

Le code suivant connecte à un serveur HTTP (*www.tcpipguide.com*) et envoie une simple requête *HEAD* pour obtenir des informations sur une ressource demandée (comme dans le premier exemple il s'agit la page d'accueil).

```
1  # Code Python 2
2  # -*- @author: Ilyas Bambrik -*-
3
4  ## Lire l'explication de DemoHTTP1_GET avant de lire ce document
5
6  # requete HEAD (obtention des informations sur le contenu d'une ressource)
7
8  import socket
9  ConnexionAuServeurHTTP=socket.socket()
10 Serveur="www.tcpipguide.com"
11 Port=80
12 ConnexionAuServeurHTTP.connect((Serveur,Port))
13
14 # l'url de cette requete = / (racine)
15 ConnexionAuServeurHTTP.send("HEAD / HTTP/1.1\r\nHost:www.tcpipguide.
   com\r\nConnection:close\r\n\r\n")
16 rec=ConnexionAuServeurHTTP.recv(1024)
17 # lire le contenu jusqu'a la fin
18 while True:
19     if len(rec)==0:
20         break
21     print rec,
22     rec=ConnexionAuServeurHTTP.recv(1024)
```

Le résultat de la requête sera la réponse suivante (entêtes descriptifs de la ressource sans la ressource elle même : taille de la ressource 20757 octets, type de la ressource texte, le serveur accepte les réponses partielles) :

```
>>>
HTTP/1.1 200 OK
Date: Thu, 13 Dec 2018 18:59:06 GMT
Server: Apache/2.4.37
Accept-Ranges: bytes
Content-Length: 20757
Content-Type: text/html

>>>
```

Réponse serveur

# 4. Entête Range

Le code suivant connecte à un serveur HTTP (*www.tcpipguide.com*) et envoie une requête GET avec l'entête *Range* pour obtenir une partie de la ressource(dans l'intervalle 500-1000).

```python
# -*- @author: Ilyas Bambrik -*-

## Lire l'explication de DemoHTTP1_GET avant de lire ce document

# Range (demande de contenu partiel) / [ reponse serveur ContentRange - Code 206 contenu partiel]

import socket
ConnexionAuServeurHTTP=socket.socket()
Serveur="www.tcpipguide.com"
Port=80
ConnexionAuServeurHTTP.connect((Serveur,Port))
# L'entete [Range: bytes=500-1000] indique que le client souhaite recevoir seulemnt les octets de 500 jusqu'a 1000
# Demande des caracteres dans l'intervale 500-1000
ConnexionAuServeurHTTP.send("GET / HTTP/1.1\r\nHost:www.tcpipguide.com\r\nRange: bytes=500-1000\r\nConnection:close\r\n\r\n")
rec=ConnexionAuServeurHTTP.recv(1024)
while True:
    if len(rec)==0:
        break
    print rec,
    rec=ConnexionAuServeurHTTP.recv(1024)

```

Le résultat de la requête sera la réponse suivante (les octets de 500 à 1000 de la page demandée) :

```
File  Edit  Shell  Debug  Options  Windows  Help
Python 2.7.9 (default, Dec 10 2014, 12:24:55) [MSC v.1500
32
Type "copyright", "credits" or "license()" for more infor
>>> ================================ RESTART ============
>>>
HTTP/1.1 206 Partial Content
Date: Thu, 13 Dec 2018 19:07:41 GMT
Server: Apache/2.4.37
Accept-Ranges: bytes
Content-Range: bytes 500-1000/20757
Content-Length: 501
Content-Type: text/html
```

Réponse du serveur

```
imate.js">
</script>
</head>

<body>

<table border="0" width="100%" cellpadding="3" cellspacin
height="522" style="border-collapse: collapse" bordercolo
  <tr>
    <td width="174" rowspan="5" valign="top">
    <!--webbot bot="Include" U-Include="side_nav.htm" TA

<div align="left">
  <table border="0" width="161" cellpadding="3" cellspaci
 height="431">
    <tr>
      <td bgcolor="#D7D7EE" align="left" width="161" vali
>>>
```

Les octets dans l'intervales 500-1000 de la ressource

# 5. Requête OPTIONS

Le code suivant connecte à un serveur HTTP (*www.tcpipguide.com*) et envoie une requête OPTIONS pour obtenir les méthodes (GET, PUT, HEAD, DELETE, etc) supportées pour la ressource désignée.

```
1  # -*- coding: cp1252 -*-
2  # OPTIONS :
3
4  import socket
5  ConnexionAuServeurHTTP=socket.socket()
6  Serveur="www.tcpipguide.com"
7  Port=80
8  ConnexionAuServeurHTTP.connect((Serveur,Port))
9  ConnexionAuServeurHTTP.send("OPTIONS / HTTP/1.1\r\nHost:www.tcpipguide.
   com\r\nConnection:close\r\n\r\n")
10
11 rec=ConnexionAuServeurHTTP.recv(1024)
12 while True:
13     if len(rec)==0:
14         break
15     print rec,
16     rec=ConnexionAuServeurHTTP.recv(1024)
```

Le résultat de la requête sera la réponse suivante (les méthodes permises sur l'url donné):

```
>>> ================================ RESTART ================================
>>>
HTTP/1.1 200 OK
Date: Thu, 13 Dec 2018 19:18:40 GMT
Server: Apache/2.4.37
Allow: POST,OPTIONS,HEAD,GET
Content-Length: 0
Content-Type: text/html

>>>
```

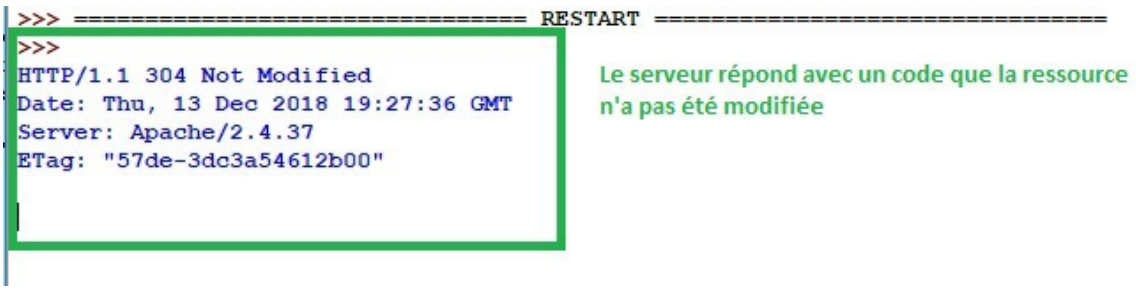
Methodes supportées

# 6. Requête If-Modified-Since

Le code suivant connecte à un serveur HTTP (*www.tcpipguide.com*) et envoie une simple requête GET pour obtenir la ressource si la ressource n'a pas été modifiée ultérieurement à la date *If-Modified-Since*.

```
1  # -*- @author: Ilyas Bambrik -*-
2
3  #Last-Modified: Mon, 07 Jun 2004 00:26:52 GMT Code reponse 304
4  import socket
5  ConnexionAuServeurHTTP=socket.socket()
6  Serveur="www.tcpipguide.com"
7  Port=80
8  ConnexionAuServeurHTTP.connect((Serveur,Port))
9  # requete avec header If-Modified-Since (telecharger la ressource si celle-ci a
   ete modifi[&e] au dela de la data specifi[&e] par If-Modified-Since
10 ConnexionAuServeurHTTP.send("GET /free/diagrams/tcpswunack.png HTTP/1.1\r\nHost:
   www.tcpipguide.com\r\nIf-Modified-Since: Mon, 07 Jun 2005 00:26:52 GMT\r\nConnection:
   close\r\n\r\n")
11
12 rec=ConnexionAuServeurHTTP.recv(1024)
13 while True:
14     if len(rec)==0:
15         break
16     print rec,
17     rec=ConnexionAuServeurHTTP.recv(1024)
18
```

Le résultat de la requête sera la réponse suivante (le serveur a répondu avec le code 304 pour dire que la ressource n'a pas été modifiée Mon, 07 Jun 2004 00:26:52 GMT):




Après l'exécution du code précédant, le serveur ne répond pas la ressource car celle-ci n'a pas été modifiée ultérieurement à la date indiqué par l'entête If-Modified-Since (Mon, 07 Jun 2005 00:26:52 GMT).

Question

Modifier la valeur de l'entête If-Modified-Since , afin que le serveur revoie la ressource demandée.

# 7. Téléchargement de fichier (png, jpeg, exe) depuis le serveur Web

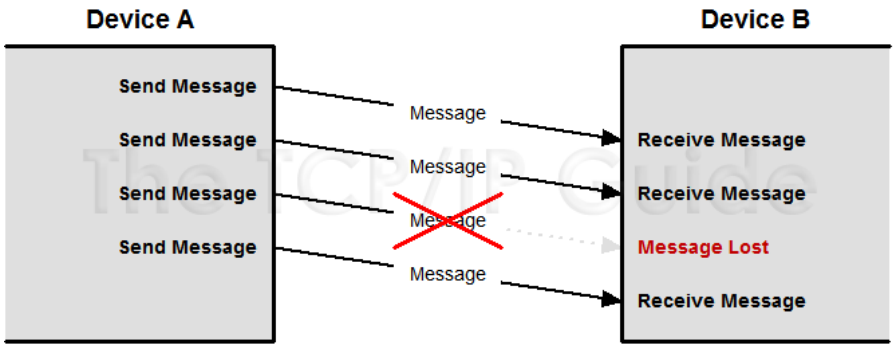
Le téléchargement de fichiers se passe exactement de la même façon que la demande de contenu HTML avec une requête GET. Cependant, pour enregistrer le contenu du fichier demandé, on doit d'abord séparer les entêtes de la réponse serveur du contenu. Pour ceci, il suffit de lire jusqu'à la séquence CR LF CR LF (*\r\n\r\n)* pour retrouver le premier octet de la ressource (ligne 16). En suite, au lieu d'afficher avec *print* les octets reçus, il suffit de d'enregistrer ceux dans un fichier.

Le code suivant envoie une requête pour le fichier *"/free/diagrams/tcpswunack.png"* (*ligne 9*) et l'enregistre dans un fichier *"tcpswunack.png"*. Après la réception du fichier complet, celui-ci est fermé avec la méthode close (*ligne 29*).

Après l'exécution du code, un fichier "tcpswunack.png" sera créé dans le même répertoire. Ouvrez celui pour voir le résultat.

```
1  # -*- @author: Ilyas Bambrik -*-
2
3  import socket
4  ConnexionAuServeurHTTP=socket.socket()
5  Serveur="www.tcpipguide.com"
6  Port=80
7  ConnexionAuServeurHTTP.connect((Serveur,Port))
8  # demande le fichier /free/diagrams/tcpswunack.png avec la methode GET
9  ConnexionAuServeurHTTP.send("GET /free/diagrams/tcpswunack.png HTTP/1.1\r\nHost:
   www.tcpipguide.com\r\nConnection:close\r\n\r\n")
10
11 rec=ConnexionAuServeurHTTP.recv(1024)
12 # rec est une chaine de caractere reçu
13 # rec contiendra les entetes HTTP de la reponse serveur ainsi que la ressource
14 # la sequence '\r\n\r\n' separe les entetes HTTP du contenu de la ressource
15 # ainsi, pour lire les caracteres de la ressources seulement, il suffit de lire
   a partir du caractere qui suit la sequence "\r\n\r\n"
16 rec=rec[rec.find("\r\n\r\n")+4::]
17
18 # ouvrir un fichier en mode ecriture en octets ("wb" == write, bytes)
19 fichier=open("tcpswunack.png","wb")
20
21 while True:
22     if len(rec)==0:
23         break
24     # au lieu d'imprimer le contnu reçu sur ecrant
25     fichier.write(rec)
26     rec=ConnexionAuServeurHTTP.recv(1024)
27
28
29 fichier.close()
30
```

L'image contenue dans *"tcpswunack.png"* est la suivante :


Device A
Device B
Send Message
Send Message
Send Message
Send Message
Message
Message
Message
Message
Receive Message
Receive Message
Message Lost
Receive Message

# 8. Téléchargement de contenu avec HTTPS

*Cette partie nécessite une connexion internet.*

La différence entre HTTP et HTTPS est le fait que dans HTTPS, tout juste après l'établissement de connexion TCP entre le client et le serveur, les deux parties s'échangent leurs certificats numériques contenant la clé publique (*RSA*). En suite, une clé privée est partagée (*AES*) entre le client et serveur.

Après la distribution des clés, les requêtes client et les réponses serveurs sont les même que HTTP mais sont chiffrées.

L'échange de certificat et les opérations de chiffrement sont assurés par le module *SSL (Secure Socket Layer)* en python (*ligne 6 et ligne 12*)

Le code suivant connecte au serveur HTTPS thescipub.com, et télécharge le fichier pdf contenu de l'url /pdf/10.3844/ajassp.2015.382.402 dans un fichier *"fichier.pdf"*:

```
1
2 # -*- coding: cp1252 -*-
3 # -*- @author: Ilyas Bambrik -*-
4 import socket
5 connexionTCP=socket.socket()
6 import ssl
7 Serveur="www.coursera.org"
8 Port=443
9
10 connexionTCP.connect((Serveur,Port))
11 ConnexionAuServeurHTTPS=ssl.create_default_context().wrap_socket(connexionTCP,
   server_hostname=Serveur)
12
13 ConnexionAuServeurHTTPS.send("GET /api/certificate.v1/pdf/XWBGNNE7HKTF HTTP/1.
   1\r\nHost:www.coursera.org\r\nConnection:close\r\n\r\n")
14 rec=""
15 while 1:
16         if "\r\n\r\n" in rec:
17                 break
18         rec+=ConnexionAuServeurHTTPS.recv(1024)
19 rec=rec[rec.find("\r\n\r\n")+4::]
20 fichier_=open("fichier.pdf","wb")
21 while True:
22         if len(rec)==0:
23                 break
24         fichier_.write(rec)
25         rec=ConnexionAuServeurHTTPS.recv(1024)
26 fichier_.close()
27 ConnexionAuServeurHTTPS.close()
28
```

# 9. requests

Afin de tester le programme suivant vous devez installer la bibliothèque *requests (https://www.agiratech.com/install-requests-library-in-python).*

```
1 import requests
2 reponse=requests.get("https://www.thescipub.com/pdf/ajassp.2015.382.402.pdf",
  stream=True)
3 fichier=open("file2.pdf","wb")
4 fichier.write(reponse.raw.data)
5 fichier.close()
6
7
```