

Le protocole HTTP

I LE PROTOCOLE HTTP COTÉ CLIENT

Le protocole de transfert hypertexte (HTTP, *Hypertext Transfer Protocol*) est un protocole de niveau application pour les systèmes d'information hypermédia distribués collaboratifs. HTTP a été utilisé par l'initiative d'informations mondiales du World-Wide Web depuis 1990. La présente spécification définit le protocole désigné sous le nom de "HTTP/1.1", qui est une mise à jour de la RFC 2068.

Les communications HTTP ont généralement lieu sur des connexions TCP/IP. L'accès par défaut est TCP 80, mais d'autres ports peuvent être utilisés. Cela n'empêche pas HTTP d'être mis en œuvre par dessus tout autre protocole sur l'Internet, ou sur d'autres réseaux. HTTP présuppose seulement un transport fiable ; tout protocole qui fournit de telles garanties peut être utilisé.

HTTP utilise un schéma de numérotation "<major>.<minor>" pour indiquer les versions du protocole. La politique de versions du protocole est destinée à permettre à l'expéditeur d'indiquer le format d'un message et sa capacité à comprendre la suite de la communication HTTP.

La version d'un message HTTP est indiquée par un champ HTTP-Version dans la première ligne du message.

Noter que les numéros majeurs et mineurs DOIVENT être traités comme des entiers séparés et que chacun PEUT être incrémenté de plus d'un chiffre.

Les URI (Uniform Resource Identifier) ont été appelés de nombreuses façons : adresses WWW, identifiants universels de documents, identifiants de ressource universels, et finalement sont la combinaison des localisateurs uniformes de ressource (URL) et des noms universels de ressource (URN). Pour ce qui concerne HTTP, les identifiants universels de ressource sont simplement des chaînes formatées qui identifient -- via le nom, la localisation, ou toute autre caractéristique -- une ressource.

Dans le protocole HTTP, une méthode est une **Commande** spécifiant un type de requête du client, c'est-à-dire qu'elle demande au serveur d'effectuer une action. En général l'action concerne une ressource identifiée par l'[URI](#) qui suit le nom de la méthode. Une requête GET est envoyée pour récupérer une page.

Par défaut, HTTP 1.1 utilise des connexions persistantes, autrement dit la connexion n'est pas immédiatement fermée après une requête, mais reste disponible pour une nouvelle requête. On appelle souvent cette fonctionnalité *keep-alive*.

Les autres paramètres de l'en-tête HTTP 1.1 sont les suivantes :

Accept

Cet en-tête liste les types MIME de contenu acceptés par le client. Le caractère étoile * peut servir à spécifier tous les types / sous-types.

Accept-Charset

Spécifie les encodages de caractères acceptés.

Accept-Language

Spécifie les langues acceptées.

I.1 EXERCICE 1 :

En analysant la capture de trames présentée ci-dessous :

Donner l'adresse IP source et l'adresse IP destination ;

Donner le port destinataire ;

Préciser si la communication a lieu en TCP ou UDP ;

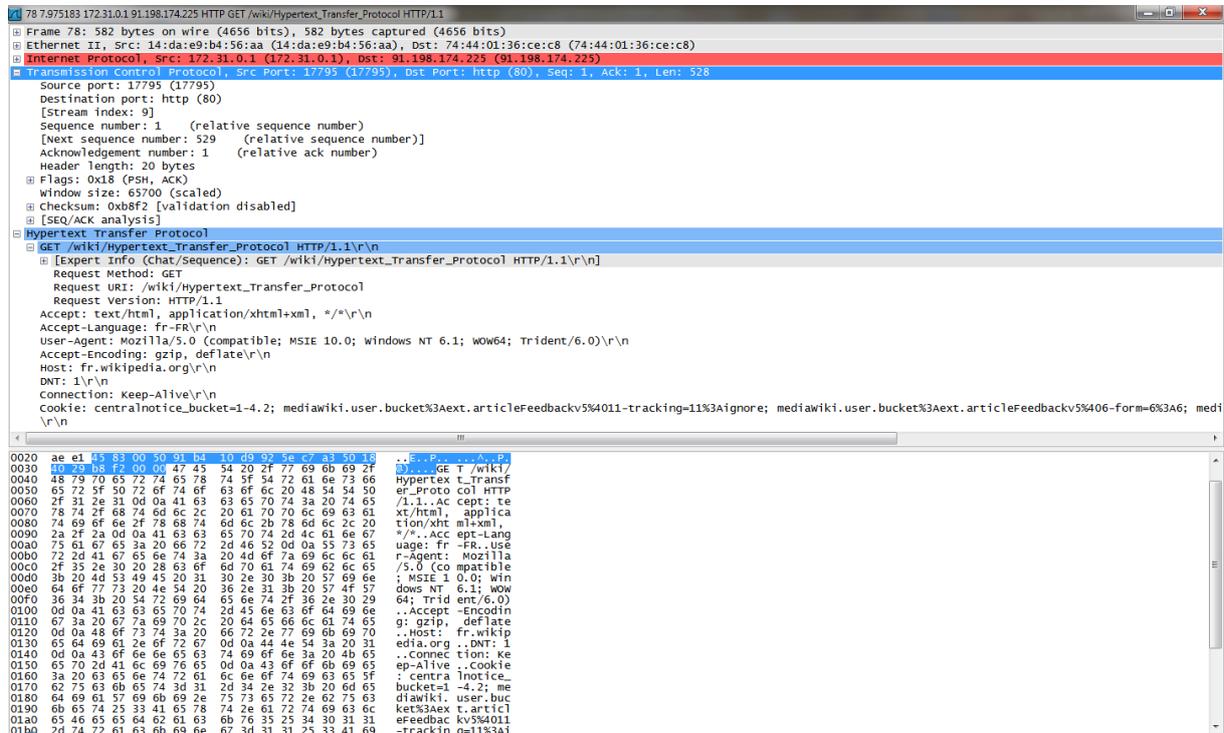
Donner la version du protocole HTTP utilisée ;

Préciser l'URI demandé ;

Donner la méthode utilisée ;

Lister les langages acceptés

Préciser si la connexion est persistante.



```
787.975183 172.31.0.1 91.198.174.225 HTTP GET /wiki/Hypertext_Transfer_Protocol HTTP/1.1
Frame 78: 582 bytes on wire (4656 bits), 582 bytes captured (4656 bits)
Ethernet II, Src: 14:da:e9:b4:56:aa (14:da:e9:b4:56:aa), Dst: 74:44:01:36:ce:c8 (74:44:01:36:ce:c8)
Internet Protocol, Src: 172.31.0.1 (172.31.0.1), Dst: 91.198.174.225 (91.198.174.225)
Transmission Control Protocol, Src Port: 17795 (17795), Dst Port: http (80), Seq: 1, Ack: 1, Len: 528
Source port: 17795 (17795)
Destination port: http (80)
[Stream index: 9]
Sequence number: 1 (relative sequence number)
[Next sequence number: 529 (relative sequence number)]
Acknowledgement number: 1 (relative ack number)
Header length: 20 bytes
Flags: 0x18 (PSH, ACK)
Window size: 65700 (scaled)
Checksum: 0xb8f2 [validation disabled]
[SEQ/ACK analysis]
Hypertext Transfer Protocol
GET /wiki/Hypertext_Transfer_Protocol HTTP/1.1\r\n
[Expert Info (Chat/Sequence): GET /wiki/Hypertext_Transfer_Protocol HTTP/1.1\r\n]
Request Method: GET
Request URI: /wiki/Hypertext_Transfer_Protocol
Request Version: HTTP/1.1
Accept: text/html,application/xhtml+xml,*/*\r\n
Accept-Language: fr-FR\r\n
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; windows NT 6.1; wow64; Trident/6.0)\r\n
Accept-Encoding: gzip, deflate\r\n
Host: fr.wikipedia.org\r\n
DNT: 1\r\n
Connection: Keep-Alive\r\n
Cookie: centralnotice_bucket-1-4.2; mediawiki.user.bucket%3Aext.articlefeedback%5%4011-tracking-11%3Aignore; mediawiki.user.bucket%3Aext.articlefeedback%5%406-form-6%3A6; mediawiki.user.bucket%3Aext.articlefeedback%5%4011-tracking-11%3Aai
0020 ae e1 45 83 00 50 91 b4 10 d9 92 5e c7 a3 50 18 ..E..P.....A..P
0030 10 29 b8 f2 00 00 47 45 54 20 2f 77 69 6b 69 2f ..GET /wiki/
0040 48 79 70 65 72 74 65 78 74 5f 54 72 61 6e 73 66 Hypertext_Transf
0050 65 72 5f 50 72 6f 74 6f 63 6f 6c 20 48 54 50 er_proto col HTTP
0060 2f 31 2e 31 0d 0a 41 63 63 65 70 74 3a 20 74 65 /1.1..Ac cept: te
0070 78 74 2f 68 74 6d 6c 2c 20 61 70 70 6c 69 63 61 xt/html, applica
0080 74 69 6f 6e 2f 78 69 74 6d 6c 2b 78 6d 6c 2c 20 tion/xht ml+xml,
0090 2a 2f 2a 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67 /*..Acc ept-Lang
00a0 75 61 67 65 3a 20 66 72 2d 46 52 0d 0a 55 73 65 uage: fr -FR..Use
00b0 72 2d 41 67 65 6e 74 3a 20 4d 6f 74 69 6c 61 r-Agent: Mozilla
00c0 2f 35 2e 30 20 28 63 6f 6d 70 61 74 69 62 6c 65 /5.0 (co mpatible
00d0 3b 20 4d 53 49 45 20 31 30 2e 30 3b 20 57 69 6e ; MSIE 1 0.0; win
00e0 64 6f 77 73 20 4e 54 20 36 2e 31 3b 20 57 4f 57 dows NT 6.1; WOW
00f0 36 34 3b 20 54 72 69 64 65 6e 74 2f 36 2e 30 29 64; Trid ent/6.0)
0100 0d 0a 41 63 63 65 70 74 2d 45 6e 63 6f 64 69 6e ..Accept -Encodin
0110 67 3a 20 67 74 69 70 2c 20 64 65 6e 6c 61 74 65 g: gzip, deflate
0120 0d 0a 48 6f 73 74 3a 20 66 72 2e 77 69 6b 69 70 ..Host: fr.wikip
0130 65 64 69 61 2e 6f 72 67 0d 0a 44 4e 54 3a 20 31 edia.org ..DNT: 1
0140 0d 0a 43 6f 6e 65 63 74 69 6f 6e 3a 20 4b 65 ..Connec tion: ke
0150 65 70 2d 41 6c 69 76 65 0d 0a 43 6f 6f 6b 69 65 ep-Alive ..cookie
0160 3a 20 63 65 6e 74 72 61 6c 6e 6f 74 69 63 65 5f ; centra lnotice_
0170 62 73 63 6b 65 74 3d 31 2d 34 2e 32 3b 20 6d 65 bucket-1 -4.2; me
0180 64 69 61 57 69 6b 69 2e 75 73 65 72 2e 62 75 63 diawiki. user.buc
0190 6b 65 74 25 33 41 65 78 74 2e 61 72 74 69 63 6c ket%3Aex t.articl
01a0 65 46 65 65 64 62 61 63 6b 76 35 25 34 30 31 31 efeedback kv%4011
01b0 2d 74 72 61 63 6b 69 6e 67 3d 31 31 25 33 41 69 -trackin nes%3Aa
```

II LES MÉTHODES DU PROTOCOLE HTTP

Il existe de nombreuses méthodes, les plus courantes étant GET, HEAD et POST :

GET

C'est la méthode la plus courante pour demander une ressource. Une requête GET est sans effet sur la ressource, il doit être possible de répéter la requête sans effet.

HEAD

Cette méthode ne demande que des informations sur la ressource, sans demander la ressource elle-même.

POST

Cette méthode est utilisée pour transmettre des données en vue d'un traitement à une ressource (le plus souvent depuis un formulaire HTML). L'URI fournie est l'URI d'une ressource à laquelle s'appliqueront les données envoyées. Le résultat peut être la création de nouvelles ressources ou la modification de ressources existantes. À cause de la mauvaise implémentation des méthodes HTTP (pour [Ajax](#)) par certains navigateurs (et la norme [HTML](#) qui ne supporte que les méthodes GET et POST pour les formulaires), cette méthode est souvent utilisée en remplacement de la requête PUT, qui devrait être utilisée pour la mise à jour de ressources.

Voici une utilisation de la méthode GET en mode console pour interroger le serveur OPENERP-WEB.

```

Telnet 172.31.0.3
<meta name="description" content="http.com" />
<meta name="robots" content="index, follow" />
<meta name="revisit-after" content="10" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<script type="text/javascript">
  document.cookie = "jsc=1";
</script>

</head>
<frameset rows="100%,*" frameborder="no" border="0" framespacing="0">
  <frame src="http://www.HTTP.com?epl=uSqzBHgplMIF6PiVOLFlrBiuarklAQuEUyU38g6KFEkfgpUUEQpo_H5LEBVvDiYovIOZ2u99WRjSFYQRFBqJiukYQNNj9aaQuM9kjhmpGcdCD1OvEmphUwocXcEp-dkzytdejqYxMI3rqqUCDNANA06inhJjUCXU011De578AA0B-AQAAQ1BbCQAahqtrE11TJ11BM1ZoUkKBAAABA" name="http.com">
</frameset>
</noframes>
<body><a href="http://www.HTTP.com?epl=uSqzBHgplMIF6PiVOLFlrBiuarklAQuEUyU38g6KFEkfgpUUEQpo_H5LEBVvDiYovIOZ2u99WRjSFYQRFBqJiukYQNNj9aaQuM9kjhmpGcdCD1OvEmphUwocXcEp-dkzytdejqYxMI3rqqUCDNANA06inhJjUCXU011De578AA0B-AQAAQ1BbCQAahqtrE11TJ11BM1ZoUkKBAAABA">Click here to go to http.com</a></body>
</noFrames>
</html>delmp@ubuntu:~$ GET http://172.31.0.3:8080 HTTP/1.1 ! more
<!doctype html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<link rel="shortcut icon" href="/openobject/static/images/favicon.ico">
<link type="text/css" rel="stylesheet" href="/openobject/static/css/jquery-ui-smoothness/jquery-ui-1.8.6.custom.css"/>
<link type="text/css" rel="stylesheet" href="/openobject/static/css/jquery.fancybox-1.3.1.css"/>
<script type="text/javascript" src="/openobject/static/javascript/MochiKit.js"></script>
<script type="text/javascript" src="/openobject/static/javascript/MochiKit/Resizable.js"></script>
<script type="text/javascript" src="/openobject/static/javascript/jquery/jquery-1.4.2.js"></script>
<script type="text/javascript" src="/openobject/static/javascript/jquery/jquery-ui-1.8.6.custom.min.js"></script>
<script type="text/javascript" src="/openobject/static/javascript/jquery/jquery.form.js"></script>
</head>
</html>
More

```

Et un autre morceau

```

Telnet 172.31.0.3
<input name="login_action" value="login" type="hidden"/>
<fieldset class="box">
  <legend style="padding: 4px;">
    
  </legend>
  <div class="box2" style="padding: 5px 5px 20px 5px">
    <table width="100%" align="center" cellpadding="2px" cellspacing="0" style="border:none;">
      <tr>
        <td class="label"><label for="db">Database:</label></td>
        <td style="padding: 3px;">
          <select name="db" id="db" class="db_user_pass">
            <option value="">
              <option value="marie" >marie
            </option>
          </select>
        </td>
      </tr>
      <tr>
        <td class="label"><label for="user">User:</label></td>
        <td style="padding: 3px;"><input type="text" id="user" name="user" class="db_user_pass" value="" autofocus="true"/></td>
      </tr>
      <tr>
        <td class="label"><label for="password">Password:</label></td>
        <td style="padding: 3px;"><input type="password" value="" id="password" name="password" class="db_user_pass"/></td>
      </tr>
      <tr>
        <td colspan="2">
          <div class="db_login_buttons">
            <button type="button" class="static_boxes" tabindex="1" onclick="location.href='/openerp/database/'>Databases
            </button>
            <button type="submit" class="static_box">Login</button>
          </div>
        </td>
      </tr>
    </table>
  </div>
</fieldset>
</form>
<div style="margin-top: 10px">
  <table cellpadding="0" cellspacing="0" width="100%" style="border:none;">
More

```

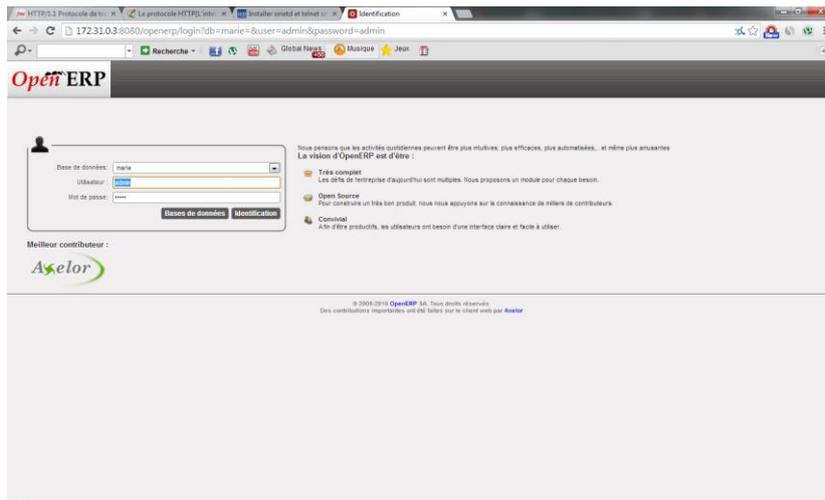
Comment travail un navigateur :

- Il appelle la page demandée GET http://172.31.0.3:8080 HTTP/1.1.
- Une fois la page reçue, il va chercher dedans tous les URI et va les appeler avec un GET. Dans le cas d'une l'image, il devra la recomposer, ce qui lui est possible parce qu'il connaît le format d'encodage gif.
- Il affiche alors la page, dans son intégralité.

La méthode GET peut transporter des paramètres (définis par l'application) pour atteindre une page précise, par exemple la page de connexion au serveur OPENERP WEB avec les paramètres "user" et "password" saisis :

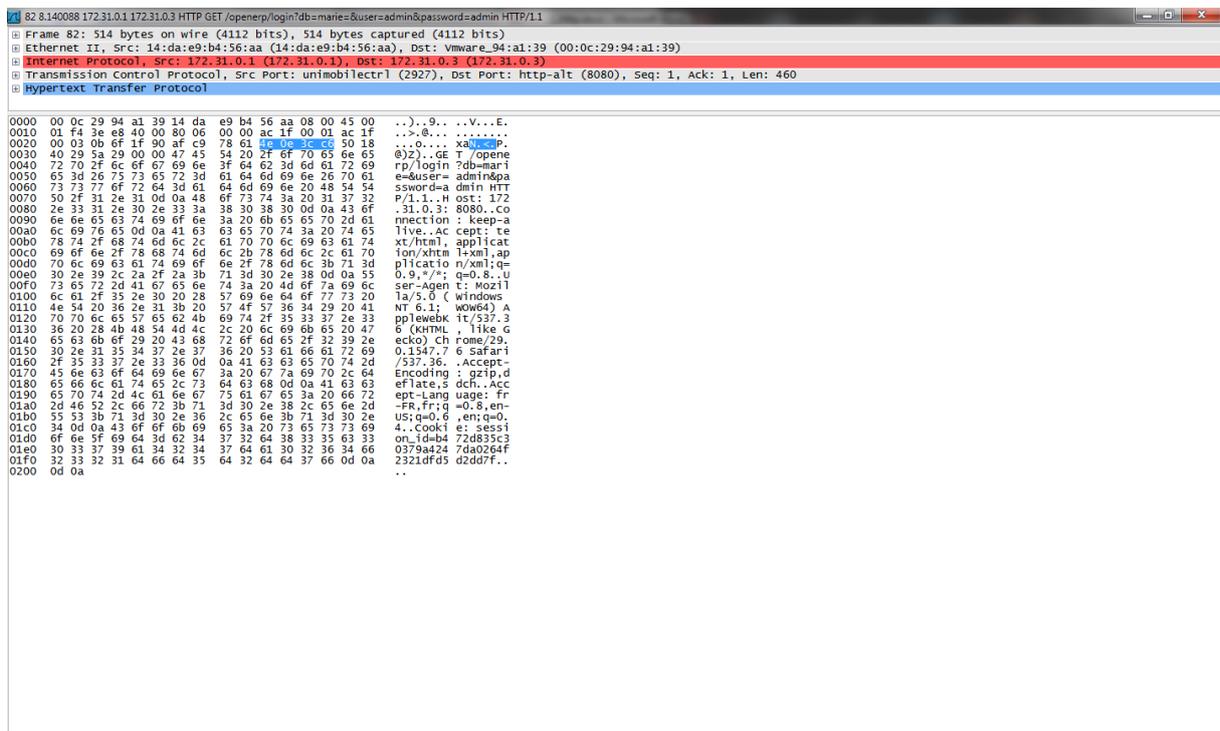
<http://172.31.0.200:8080/openerp/login?db=&user=admin&password=admin>

Réponse obtenue dans un navigateur:



II.1 EXERCICE 2 :

En analysant la capture de trames présentées ci-dessous :



Donner l'adresse du serveur interrogé ;

Donner le nom de la page demandée ;

Préciser si des paramètres ont été fournis ;

Conclure sur les dangers d'un tel passage de paramètres.