

10.5

Requête HTTP

NSI 1ÈRE - JB DUTHOIT

10.5.1 Côté client, côté serveur

L'objectif est ici de voir comment circulent les informations sur le réseau internet.

Imaginons un client (un ordinateur personnel par exemple) qui demande avec son navigateur une page Web. Il s'agit d'une requête. Un serveur (qui est un ordinateur aussi) répond aux demandes lorsqu'il est interrogé.

La demande et la réponse se font grâce à un protocole de communication, le protocole **HTTP**.

Nous allons ici étudier cette communication.

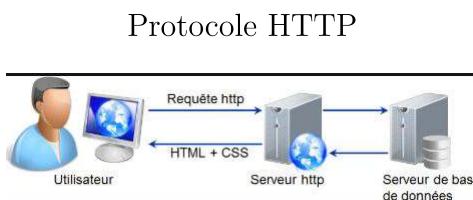
On peut ainsi distinguer :

- Le côté client ou front-end :

- Il est centré sur le navigateur, et il est constitué par de pages HTML, CSS et du code JS.
- Les métiers concernés sont par exemple web designer, intégrateur CSS, ou développeur front-end.
- certaines vérifications de formulaires sont faites dans le navigateur via des attributs HTML spécialisés, comme pattern (ou avec du code JS).

- Le côté serveur ou back-end :

- Il assure la réponse à une demande faite lors de la soumission du formulaire par validation (submit ou avec un appel ajax en JS.)
- On utilise pour cela plusieurs outils et langage : PHP, Pythonet les bases de données relationnelles (programme de terminale).
- Ces programmes s'exécutent le plus souvent sur des serveurs distants, sur internet.
- Des vérifications de données sont aussi réalisées sur le serveur, notamment pour des questions de sécurité.



C'est donc le rôle de protocole HTTP que d'assurer le dialogue entre serveur et client. HTTP signifie HyperText Transfer Protocol. La transmission de paramètres (champs saisis dans un formulaire) se fait grâce à ce protocole.

Remarque

Pour davantage de sécurité, on ajoute une couche **SSL** (Secure Socket Layer) à HTTP, lequel devient alors **HTTPS**. Cela assure le chiffrement des données et donc la confidentialité des échanges

10.5.2 Transmission des données

Les protocoles TCP et IP permet, on le rappelle, à tout appareil connecté de dialoguer avec une autre machine connectée.

Le principe du protocole TCP/IP est similaire à celui d'une expédition de colis par la poste. Si je dois envoyer un meuble en kit, il faut commencer par le démonter et d'emballer chaque pièce. Ensuite, il faut porter chaque paquet au bureau de poste le plus proche. Celui-ci va transmettre ces paquets au bureau de poste principal. Les paquets pourront prendre ensuite des chemins différents (suivant la taille, les disponibilités des trains, avions ...) Comme la notice finale est jointe à chaque paquet, on pourra vérifier qu'il ne manque pas de colis (et faire le cas échéant une réclamation), et il sera aisément de monter le meuble.

C'est la même chose sur internet ! Si j'envoie un message, celui-ci est décomposé en petits paquets (qui contiennent des blocs de données). C'est le rôle du protocole TCP. Les différents paquets vont transiter par des routeurs différents jusqu'à la destination finale. Chaque paquet peut avoir une route différente. C'est le rôle du protocole IP de les faire arriver au bon destinataire, et celui-ci pourra connaître l'expéditeur. À l'arrivée, l'ensemble des paquets permettent de reconstituer le message envoyé, grâce au protocole TCP.

Ainsi, et pour simplifier :

- Le protocole TCP se charge de découper le message en petits paquets et au final de reconstituer le message
- Le protocole IP ajoute au paquets l'adresse du destinataire et de l'expéditeur. Cela permet au destinataire d'envoyer des accusés de réception pour chaque paquet. L'expéditeur peut éventuellement renvoyer des paquets qui se seraient perdus.

Le protocole HTTP est utilisé au-dessus. Pour sécuriser le transport, on remplace HTTP par HTTPS ; celui-ci crée une sorte de tunnel virtuel qui crypte les données.

10.5.3 Comment obtenir l'adresse IP d'un site ?

On saisit l'adresse d'une page Web dans un navigateur :

"[qwant.com](http://www.qwant.com)" ou "<https://www.qwant.com>".

qwant.com est appelé nom de domaine.

Remarque

Pour connaître l'adresse IP d'un site, il suffit de taper dans une fenêtre de commande : ping google.fr :

```
ping www.google.fr
```

```
C:\>ping google.fr

Envoi d'une requête 'ping' sur google.fr [172.217.17.35] avec 32 octets de données :
Réponse de 172.217.17.35 : octets=32 temps=18 ms TTL=51
Réponse de 172.217.17.35 : octets=32 temps=17 ms TTL=51
Réponse de 172.217.17.35 : octets=32 temps=18 ms TTL=51
Réponse de 172.217.17.35 : octets=32 temps=18 ms TTL=51

Statistiques Ping pour 172.217.17.35:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 17ms, Maximum = 18ms, Moyenne = 17ms
```

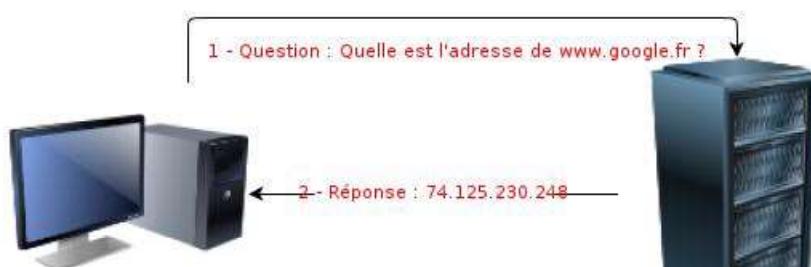
Définition

un serveur DNS est un annuaire pour ordinateur. Lorsque vous voulez accéder à un ordinateur dans le réseau, votre ordinateur va interroger le serveur DNS pour récupérer l'adresse de l'ordinateur que vous voulez joindre. Une fois, que votre ordinateur aura récupéré l'adresse du destinataire, il pourra le joindre directement avec son adresse IP.

Le serveur DNS va permettre de faire la relation entre nom d'ordinateur et adresse IP.

Une requête est alors envoyée avec le protocole DNS (Domain Name System) pour obtenir l'adresse IP du site.

requête DNS



Pour le transport, c'est le protocole UDP² qui est utilisé.

- le protocole DNS contient des données (www.qwant.fr) sur 32 octets.
- Pour le transport, le protocole UDP utilise 8 octets pour les ports sources et destinations.

2. User Datagram Protocol (UDP) est un protocole de couche transport d'internet qui permet la transmission de données sous forme de datagrammes de manière très simple.

Contrairement à TCP, avec UDP, aucune communication préalable n'est requise pour établir la connexion. UDP utilise un mode de transmission sans connexion.

Avec UDP, il n'y a pas de garantie quant à la livraison et l'ordre d'arrivée des datagrammes. UDP est adapté lorsque la détection et la correction d'erreurs ne sont pas nécessaires, ou sont effectuées directement par l'application. Le protocole UDP est utile pour transmettre rapidement de petites quantités de données, depuis un serveur vers de nombreux clients ou bien dans des cas où la perte d'un datagramme est préférée à l'attente de sa retransmission. La voix sur IP ou les jeux en ligne utilisent souvent ce protocole.

- Ensuite, le protocole IP ajoute la version, l'adresse du destinataire, de l'expéditeur, sur 20 octets.
- Enfin, le protocole Ethernet complète en avec les adresses MAC du destinataire et de la source, ainsi que le type d'IP (v4 ou v6), soit 14 octets.

On parle d'encapsulation. La réponse contient les mêmes informations avec en plus à la fin l'adresse IP demandé (4 octets)

10.5.4 Comment obtenir la page d'accueil du site ?

Les protocoles utilisés sont ici HTTP, TCP et IP et Ethernet.

Maintenant que l'ordinateur dispose de l'adresse IP du site, il peut envoyer la demande de page d'accueil du site.

Il s'agit d'une requête HTTP, et il commence par GET/HTTP/1.1.

TCP est chargé de la constitution des paquets . La taille maxi de chaque paquet est de 1500 octets (dont 40 octets pour l'en-tête). TCP s'occupe de la partie transport. A l'arrivée, les paquets sont assemblés par le TCP pour constituer par exemple un fichier HTML.

En réponse, le serveur envoie donc HTTP/1.1 200 OK, suivi par tout le code HTML de la page.

Le code 200 signifie que tout fonctionne correctement³.

Le navigateur interprète ensuite les fichiers et affiche la page Web.

A la lecture du code HTML, il peut effectuer d'autres requêtes, comme par exemple la demande de lire un fichier CSS : GET/styles/index.css HTTP/1.1, avec en réponse HTTP/1.1 200 OK (text.css) et le contenu du fichier CSS.

Il peut s'agir aussi d'images : GET/img/png/menu.png HTTP/1.1, avec une réponse HTTP/1.1 200 OK (PNG).

Le navigateur affiche un à un les éléments en fonctions des réponses reçues.

Un site peut être statique, et alors c'est la même page qui est envoyée à tous les clients qui le demandent.

Un site peut être dynamique , et dans ce cas, une page différente peut être conçue pour chaque client.

Le navigateur enregistre des copies pages consultées sur le disque dur. Cela permet un gain de temps si une page statique est demandée à nouveau.

10.5.5 Illustration avec un programme Python

Voici un programme Python nommé serveur.py qui écoute sur un port et renvoie une page HTML.

```
from socket import socket, AF_INET, SOCK_STREAM
```

3. Le code 404 correspond à l'erreur "page non trouvée"

```

reponse = b"""HTTP/1.1 200 OK
host: mon site local
Content-Type: text/html \n
<!DOCTYPE html PUBLIC>
<html>
<head>
<title> Ma page </title>
</head>
<body>
<h1> Bonjour </h1>
<h2> Le test est concluant !! </h2>
</body>
</html>"""

s = socket(AF_INET, SOCK_STREAM)
s.bind(("" ,13450)) # le programme python \'ecoute sur le port
13450

s.listen(5) # 5 connexions maximales
while True:
    connexion, adresse = s.accept()
    print('Connexion de',adresse)
    req = connexion.recv(1024).decode()
    if req != "":
        print(req)
        connexion.send(reponse)
    connexion.close()

```

Il faut exécuter ce programme et le curseur de l'interpréteur python clignote (il est en attente). Sur un navigateur, et sur le même ordinateur, nous écrivons dans la barre d'adresse 127.0.0.1 :13450.

La page Web s'affiche alors !

10.5.6 Méthodes GET et POST

La méthode GET :

- Il s'agit de la méthode la plus fréquente.
- Une requête GET n'a aucun effet sur la ressource.
- Avec cette méthode, les données du formulaires seront encodées dans une URL construite par le navigateur du client et transmise au serveur.
- C'est ce que fait par exemple Google⁶ lors d'une recherche web.
- La longueur de l'URL est parfois limitée ; cette méthode fonctionne donc avec des formulaires de taille raisonnable.
- Avec cette méthode, les données sont visibles dans l'URL ; elle n'est donc pas adaptée pour des données sensibles.

La méthode POST

- Les paramètres sont transmis au programme externe et sont invisibles dans l'URL.
- C'est une méthode plus compliquée que la méthode GET

10.5.7 Exemple de formulaire HTML, avec la méthode GET ,et une action du type HTML.

Avant d'exécuter ce fichier HTML, il est nécessaire d'exécuter le fichier serveur.py .

Voici le code du petit formulaire :

```

1 <html>
2     <head>
3         <title> Formulaire </title>
4     </head>
5     <body>
6         <p> Voici un formulaire </p>
7         <form method="GET" action = "http://127.0.0.1:13450">
8             <p>
9                 <label for="nom">Nom</label>
10                <input type="text" name="utilisateur" id="nom" autofocus/>
11            </p>
12            <input type="submit" value = "Envoyer" />
13        </form>
14    </body>
15 </html>
```

Le formulaire HTML et le champ renseigné

Voici un formulaire

Nom

Le retour du "serveur" et le champ renseigné se retrouve en URL

127.0.0.1:13450/?utilisateur=JBaptiste+Duthoit

10.5.8 Exemple de formulaire avec la méthode POST ,et une action du type HTML.

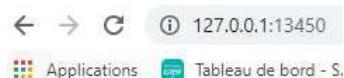
Avant d'exécuter ce fichier HTML, il est nécessaire d'exécuter le fichier serveur.py .

Voici le code du formulaire :

```

1 <html>
2     <head>
3         <title> Formulaire </title>
4     </head>
5     <body>
6         <p> Voici un formulaire </p>
7         <form method="POST" action = "http://127.0.0.1:13450">
8             <p>
9                 <label for="nom">Nom</label>
10                <input type="text" name="utilisateur" id="nom" autofocus/>
11            </p>
12            <input type="submit" value = "Envoyer" />
13        </form>
14    </body>
15 </html>
```

Pour le même champ saisi, on ne retrouve plus l'information dans l'URL



Bonjour

Le test est concluant !!

Les données sont insérées dans le corps de la requête

```
utilisateur=JBaptiste+Duthoit
Connexion de ('127.0.0.1', 61563)
GET /favicon.ico HTTP/1.1
Host: 127.0.0.1:13450
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537
.36 (KHTML, like Gecko) Chrome/80.0.3987.149 Safari/537.36
Sec-Fetch-Dest: image
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Referer: http://127.0.0.1:13450/
Accept-Encoding: gzip, deflate, br
Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7
```

10.5.9 un exemple de formulaire en HTML avec action PHP

Voici le code d'un petit formulaire, avec une action PHP :

```
<html>
  <head>
    <title> Formulaire </title>
  </head>
  <body>
    <p> Voici un formulaire </p>
    <form method="GET" action = "reponse.php">
      <p>
        <label for="nom">Nom</label>
        <input type="text" title="Entrer votre nom" pattern="[^a-zA-Z ]{2,20}" name="utilisateur" id="nom" required
placeholder="Prénom Nom"/>
      </p>
      <input type="submit" value = "Envoyer"/>
    </form>
  </body>
</html>
```

- pattern permet de vérifier que la valeur entrée est bien la valeur attendue
- required pour signifier que le champ est obligatoire
- placeholder pour donner un exemple

Remarque

| La soumission du formulaire envoie une erreur car le fichier PHP est inexistant...Ce qui nous amène au point suivant :-)