PROTOCOLE HTTP

1 – Historique

HTTP a été inventé par Tim Berners-Lee avec les adresses Web et le langage HTML pour créer le World Wide Web. À cette époque le File Transfer Protocol (FTP) était déjà disponible pour transférer des fichiers, mais il ne supportait pas la notion de format de données telle qu'introduite par Multipurpose Internet Mail Extensions (MIME). La première version de HTTP était très basique, mais prévoyait déjà le support d'entêtes MIME pour décrire les données transmises. Cette première version reste encore partiellement utilisable en 2007, connue sous le nom de HTTP/0.9.

En mai 1996, HTTP/1.0 devient finalement standard de l'IETF et est décrit dans la RFC 1945. Cette version supporte les serveurs HTTP virtuels, la gestion de cache et l'identification.

En janvier 1997 HTTP/1.1 est décrit dans la RFC 2068 de l'IETF, puis dans la RFC 2616 en juin 1999. Cette version ajoute le support du transfert en *pipeline* et la négociation de type de contenu (format de données, langue).

2 - Du client au serveur

La liaison entre le client et le serveur n'est pas toujours directe, il peut exister des machines intermédiaires servant de relais :

- Un serveur mandataire (ou *proxy*) peut modifier les réponses et requêtes qu'il reçoit et peut gérer un cache des ressources demandées.
- Une passerelle (ou *gateway*) est un intermédiaire modifiant le protocole utilisé.
- Un tunnel transmet les requêtes et les réponses sans aucune modification, ni mise en cache.

3 - Méthodes

Dans le protocole HTTP, une méthode est une **Commande** spécifiant un type de requête, c'est-à-dire qu'elle demande au serveur d'effectuer une action. En général l'action concerne une ressource identifiée par l'URL qui suit le nom de la méthode.

`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 doit être utilisée pour ajouter une nouvelle ressource (un message sur un forum ou un article dans un site). L'URI fournie est

	l'URI d'une ressource liée à la nouvelle ressource (comme l'URI du forum ou site) et non l'URI de la ressource nouvellement crée.
`OPTIONS`	Cette méthode permet d'obtenir les options de communication d'une ressource ou du serveur en général.
`CONNECT`	Cette méthode permet d'utiliser un proxy comme un tunnel de communication.
`TRACE`	Cette méthode demande au serveur de retourner ce qu'il a reçu, dans le but de tester et effectuer un diagnostic sur la connexion.
`PUT`	Cette méthode permet de remplacer ou d'ajouter une ressource sur le serveur. L'URI fourni est celui de la ressource en question.
`DELETE`	Cette méthode permet de supprimer une ressource du serveur.

Ces 2 dernières méthodes nécessitent généralement un accès privilégié.

4 - Identification

HTTP permet l'identification du visiteur par transmission d'un nom et d'un mot de passe. Il existe 2 modes d'identification : *Basic* et *Digest* (RFC 2617). Le premier mode transmet le mot de passe en clair, et ne doit donc être utilisé qu'avec le protocole HTTPS. Le deuxième mode permet une identification sans transmettre le mot de passe en clair. L'identification est cependant souvent effectuée par une couche applicative supérieure à HTTP.

5 - HTTP 1.0

Le protocole HTTP 1.0, décrit dans le RFC 1945, prévoit l'utilisation d'en-têtes inspirés de MIME. La gestion de la connexion reste identique à HTTP 0.9 : le client établit la connexion, envoie une requête, le serveur répond et ferme immédiatement la connexion.

Une requête HTTP présente le format suivant :

```
Ligne de commande (Commande, URL, Version de protocole)
En-tête de requête
[Ligne vide]
Corps de requête
```

Les réponses HTTP présentent le format suivant :

```
   Ligne de statut (Version, Code-réponse, Texte-réponse)
En-tête de réponse
[Ligne vide]
Corps de réponse
```

Requête :

```
GET /page.html HTTP/1.0
Host: example.com
Referer: http://example.com/
User-Agent: CERN-LineMode/2.15 libwww/2.17b3
```

La version du protocole HTTP est précisée suite à l'URI. La requête doit être terminée par un double retour à la ligne (CRLFCRLF). HTTP 1.0 supporte aussi les méthodes HEAD et POST. On constate l'usage d'en-têtes inspirés de MIME pour transférer les méta-données :

Host	Permet de préciser le site Web concerné par la requête, ce qui est nécessaire pour un serveur hébergeant plusieurs sites à la même adresse IP (*name based virtual host*, hôte virtuel basé sur le nom). C'est le seul en-tête réellement important.
Referer	Indique l'URI du document qui a donné un lien sur la ressource demandée. Cet en-tête permet aux webmasters d'observer d'où viennent les visiteurs.
User-Agent	Indique le logiciel utilisé pour se connecter. Il s'agit généralement d'un navigateur Web ou d'un robot d'indexation.

Réponse :

```
HTTP/1.0 200 OK
Date: Fri, 31 Dec 1999 23:59:59 GMT
Server: Apache/0.8.4
Content-Type: text/html
Content-Length: 59
Expires: Sat, 01 Jan 2000 00:59:59 GMT
Last-modified: Fri, 09 Aug 1996 14:21:40 GMT
```

```
<HTML><HEAD>
<TITLE>Example</TITLE>
</HEAD>
<BODY>
<P>Ceci est une page d'exemple.</P>
</BODY>
</HTML>
```

La première ligne donne le code de statut HTTP (200 dans ce cas).

`Date`	Moment auquel le message est généré.
`Server`	Indique quel modèle de serveur HTTP répond à la requête.
`Content-Length`	Indique la taille en octets de la ressource.
`Content-Type`	Indique le type MIME de la ressource.
`Expires`	Indique le moment après lequel la ressource devrait être considérée obsolète ; permet aux navigateurs Web de déterminer jusqu'à quand garder la ressource en mémoire cache.
`Last-Modified`	Indique la date de dernière modification de la ressource demandée.

6 - HTTP 1.1

Le protocole HTTP 1.1 est décrit par le RFC 2616 qui rend le RFC 2068 obsolète. La différence avec HTTP 1.0 est une meilleure gestion du cache. L'en-tête **Host** devient obligatoire dans les requêtes.

Les soucis majeurs des deux premières versions du protocole HTTP sont d'une part le nombre important de connexions lors du chargement d'une page complexe (contenant beaucoup d'images ou d'animations) et d'autre part le temps d'ouverture d'une connexion entre client et serveur (l'établissement d'une connexion TCP prend un temps triple de la latence entre client et serveur). Des expérimentations de connexions persistantes ont cependant été effectuées avec HTTP 1.0 (notamment par l'emploi de l'en-tête **Connection: Keep-Alive**), mais cela n'a été définitivement mis au point qu'avec HTTP 1.1.

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*. Il est aussi permis à un client HTTP d'envoyer plusieurs requêtes sur la même connexion sans attendre les réponses. On appelle cette fonctionnalité *pipelining*. La persistance des connexions permet d'accélérer le chargement de pages contenant plusieurs ressources, tout en diminuant la charge du réseau.

La gestion de la persistance d'une connexion est gérée par l'en-tête **Connection**.

HTTP 1.1 supporte la négociation de contenu. Un client HTTP 1.1 peut accompagner la requête pour une ressource d'en-têtes indiquant quels sont les langues et formats de données préférés. Il s'agit des en-têtes dont le nom commence par **Accept-**.

Les en-têtes supplémentaires supportés par HTTP 1.1 sont :

`Connection`	Cet en-tête peut être envoyé par le client ou le serveur et contient une liste de nom spécifiant les options à utiliser avec la connexion actuelle. Si une option possède des paramètres ceux-ci sont spécifiés par l'en-tête portant le même nom que l'option (**Keep-Alive** par exemple, pour spécifier le nombre maximum de requêtes par connexion). Le nom **close** est reservé pour spécifier que la connexion doit être fermée après traitement de la requête en cours.
`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 langages acceptés.

L'ordre de préférence de chaque option (type, encodage ou langage) est spécifié par le paramètre optionnel **q** contenant une valeur décimale entre 0 (*inacceptable*) et 1 (*acceptable*) inclus (3 décimales maximum après la virgule), valant 1 par défaut.

Le support des connexions persistantes doit également fonctionner dans les cas où la taille de la ressource n'est pas connue d'avance (ressource générée dynamiquement par le serveur, flux externe au serveur, ...).

Pour cela, l'encodage de transfert nommé **chunked** permet de transmettre la ressource par morceaux consécutifs en précédant chacun par une ligne de texte donnant la taille de celui-ci en hexadécimal. Le transfert se termine alors par un morceau de taille nulle, où des en-têtes finaux peuvent être envoyés.