Chapitre IV Partie 4 Hypertext Transfer Protocol (HTTP)

Ilyas Bambrik

Table des matières

Hypertext Transfer Protocol (HTTP)

I

- Actuellement le World Wide Web (WWW) est l'application la plus utilisée sur Internet (et la plus influente sur le monde) ;
- Le Web est une application tellement complexe et qui continue d'évoluer depuis ça conception ;
- Si *HTML* (*HyperText Markup Language*) est le langage de conception du contenu Web (documents hypertext), *HTTP* (*Hypertext Transfer Protocol*) est le protocole de la couche application permettant le transfert du contenu Web ;
- *Uniform Resource Locators* (*URLs*) est la méthode permettant d'identifier et référencer les ressources sur le Web d'une manière unique.

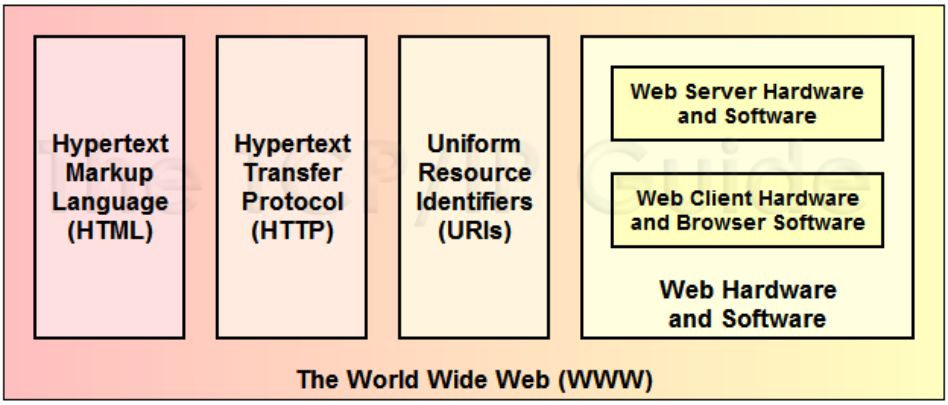


Figure 1 Composant du WWW [tcpipguide.com]

Syntaxe URL HTTP

II

La syntaxe générale d'un URL HTTP *absolu*:

http://<host>:<port>/<chemin-url>?<query>#<bookmark>

- <*host*> : Nom de domaine ou adresse IP du serveur Web où la ressource est située ;
- <*port*> : Numéro de port TCP pour connecter au serveur Web. Si le port n'est pas spécifié, le port 80 sera utilisé ;
- <*chemin-url*> : Chemin qui pointe sur la ressource demandée ;
- <*query*> : Requête optionnelle ;
- <*bookmark*> : Permet d'accéder à une section spécifique de la page Web directement ;

La forme générale d'un URL est :

http://<host>/<chemin-url>

Par exemple :

https://www.eurosport.com/football/

Remarque

- Quand le numéro de port est spécifié, il est fortement possible que plusieurs serveurs Web existent sur la même adresse IP (le serveur Web utilise un numéro de port distinct pour chaque site hébergée) ;
- Les noms de domaines ne sont pas sensibles à la case. Par contre, les URI sont sensibles à la case.

Versions HTTP

III

- *HTTP / 0.9 :* A été simplement conçu pour la transmission de documents hypertext seulement (aucun autre média supporté) ;
- *HTTP / 1.0 :* Introduit le format des messages client et les formats des réponses serveur. Cette version a étendu le contenu transporté par le protocole vers d'autres types de médias (les images surtout) ;
- *HTTP / 1.1 :* Les versions précédentes nécessitent l'initiation de session TCP pour chaque requête et la fermeture de la connexion après la livraison, ce qui diminue la performance. Ce problème a été résolu dans la version 1.1 par la transmission plusieurs requêtes sur la même session. Cette version a aussi introduit les requêtes partielles *(entête Range :)* et d'autres fonctionnalités;
- *HTTP / 2.0* (*2015*): Compression des entêtes HTTP, et multiplexage des ressources transmis ;

Remarque

Si la version 0.9 ou 1.0 est spécifiée dans la requête HTTP, le serveur fermera automatiquement la connexion TCP après réponse.

Foncionnement HTTP

IV

- HTTP fonctionne d'une manière plus simple que FTP et SMTP.
- Une connexion TCP est établie depuis le client vers le port 80 du serveur initialement (*si un autre port que TCP 80 est utilisé par le serveur, celui la doit être spécifié dans l'URL*);
- Le client transmet ses requêtes HTTP sur la connexion ;
- Pour chaque requête reçue, le serveur répond par une ou plusieurs réponses HTTP (comme dans FTP il peut y avoir des réponses intermédiaires pour une requête);
- *Caching et proxies :* Comme dans le cours DNS, le caching des ressources récemment récupérées est employé pour répondre plus rapidement au requêtes de l'utilisateur ;

Requêtes et Réponses HTTP

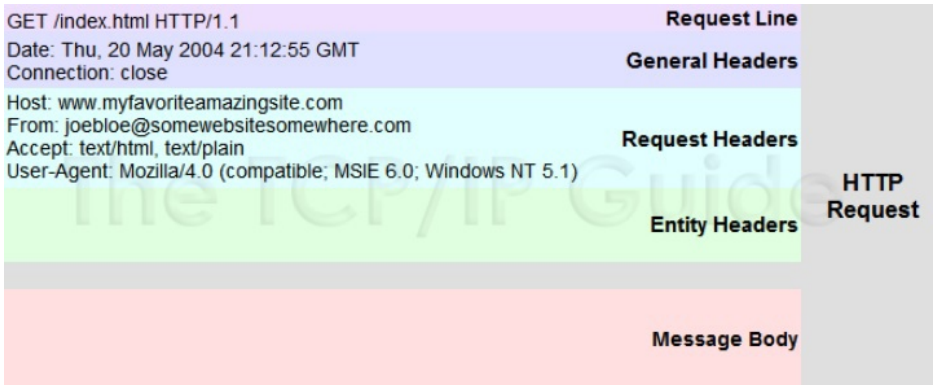
V

Syntaxe : Requête HTTP

Le format générale d'une requête HTTP est le suivant :

```
<request-line>
<general-headers>
<request-headers>
<entity-headers>
<empty-line>
[<message-body>]
[<message-trailers>]
```

- Seul <*request-line*> et le *request-header Host* sont obligatoires dans une requête HTTP;
- <*request-line*> : Indique l'action (dite Méthode) souhaitée par le client (généralement GET), l'URI de la ressource souhaitée, et la version de HTTP supportée par le client ;
- Généralement une requête ne contient pas la partie <message-body> (sauf si l'utilisateur souhaite transmettre quelque chose au serveur) ;



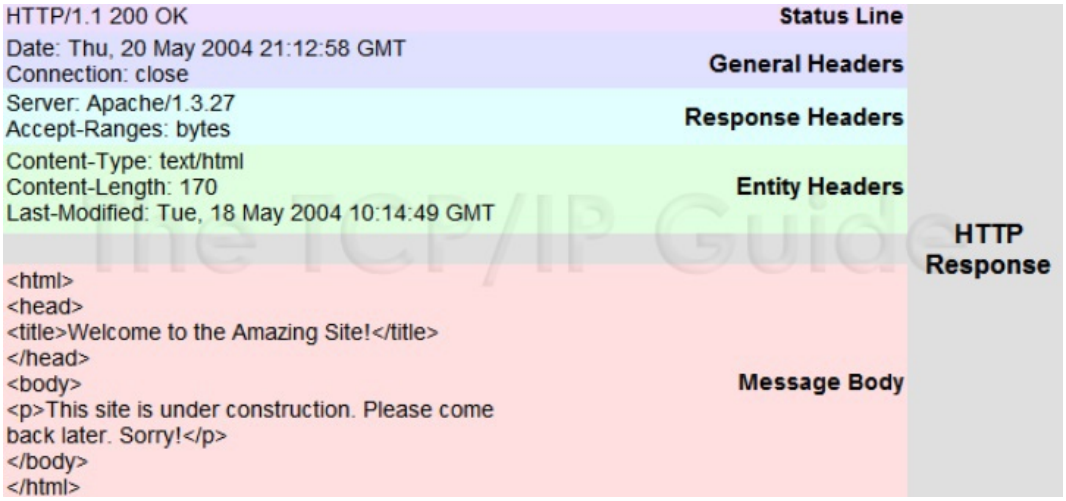
Syntaxe : Réponse HTTP

Le format générale d'une réponse HTTP est le suivant :

```
<status-line>
<general-headers>
<response-headers>
```

```
<entity-headers>
<empty-line>
[<message-body>]
[<message-trailers>]
```

- <*status-line*> Indique la version utilisée par le serveur et un code qui représente le statut du traitement de la requête par le serveur (*les codes de réponses sont structurés d'une manière similaire à ceux de FTP*) et un message descriptif;
- <*Entity-headers*> : Contient des informations concernant le contenu transmis par le serveur (le nombre de caractères dans la page renvoyée par exemple);
- *Message Body* contient généralement la ressource demandée par le client (si la requête client n'a pas pu être servie, Message Body contiendra une page indiquant l'erreur qui c'est produite comme erreur 404 ou 403) ;



Fondamental

- Chaque entête dans un message HTTP se termine par *CRLF* (comme dans TELNET et FTP) ;
- Dans une réponse HTTP serveur, une séquence *CRLFCRLF* (*double CRLF*) marque la fin des entêtes et l début du *Message Body* (la ressource que le client a demandé) ;
- Dans une requête client,*CRLFCRLF* indique la fin des entêtes HTTP;

Méthodes HTTP

VI

- *GET* : Permet d'obtenir la ressource spécifiée par l'URI. Si la ressource désignée par l'URI est accessible, le serveur revoie la ressource dans la partie Message Body de la réponse. Sinon (l'URI est incorrecte), le serveur revoie un code d'erreur (404 par exemple) dans la réponse et un message descriptive. Si la requête GET contient un header *If-Modified-Since*, la ressource est retournée seulement si la ressource a été modifiée après la date indiquée par cet entête (Sinon, le serveur renvoie un code 304 sans la ressource demandée) ;
- *HEAD* : Similaire à GET mais demande l'obtention de la réponse serveur sans le contenu de la ressource (le serveur renvoie seulement la partie <entity headers> pour que le client apprend la description de la ressource demandée) ;
- *POST* : Permet la soumission des informations par le client à un programme dans le serveur afin que ces derniers soit traités ;
- *OPTIONS* : Permet d'obtenir les options de communication supportées par le serveur. Si le paramètre de cette méthode est un URI d'une ressource, la réponse sera relative à la ressource elle même. Sinon, le caractère * peut être utilisé pour demander des informations sur le serveur lui même ;
- *PUT :* Permet de transmettre le contenu d'une ressource depuis le client vers le serveur (l'inverse de GET)
- *DELETE* permet simplement de supprimer une ressource et *TRACE* permet de recevoir la requête transmis initiale générée par le client ;

Codes de réponses

VII

Les codes des réponses serveurs sont similaires à ceux de FTP. Par contre, seul le premier chiffre est utilisé pour répartir les réponses sur 5 catégories et les deux derniers chiffres sont utilisés pour avoir 100 codes pour chaque catégorie:

- *1XX* : Message d'information général ;
- *2XX* : Requête exécuté avec succès;
- *3XX* : La requête nécessite d'autres actions pour être accomplie ;
- *4XX* : Requête du client est invalide ;
- *5XX* : Le serveur n'a pas été capable d'exécuter la requête ;

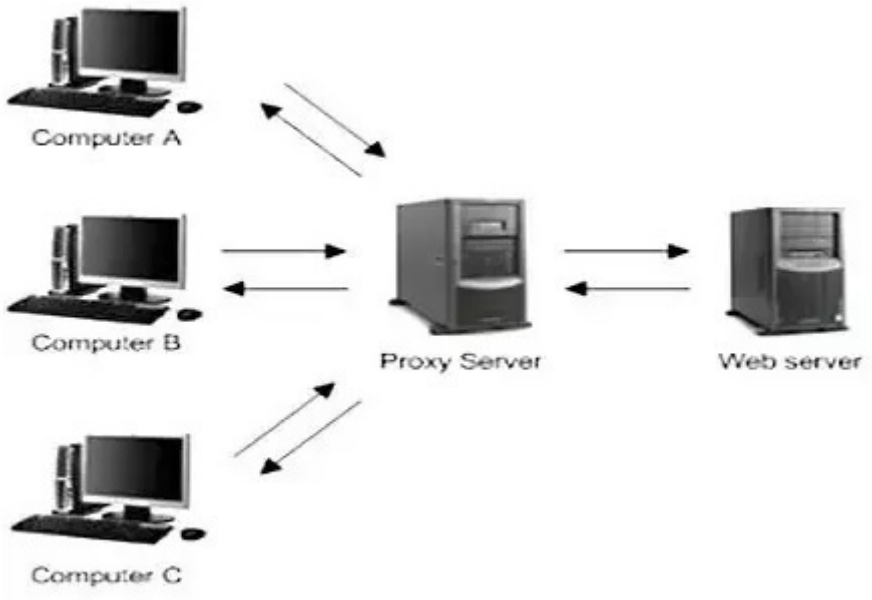
Exemple : Exemples de code de réponses

- *100* : Le client peut continuer à transmettre ses requêtes ;
- *200* : La requête a été exécutée avec succès ;
- *206* : La réponse du serveur contient le contenu partiel de la ressource demandée;
- *307* : La ressource demandée est temporairement localisée dans un URL différent ;
- *404* : URL invalide ou la ressource a été supprimée ;
- *505* : Erreur indiquant que le serveur ne supporte pas la version HTTP que le client souhaite utiliser ;

Caching et proxies

VIII

- Un proxy n'est qu'une machine intermédiaire qui permet d'effectuer la connexion avec le serveur à la place des clients ;
- Essentiellement, le proxy transmet la requête client à la place du client et retransmet les résultats reçus pa le serveur vers le client. Au cours de ce processus, le proxy mémorise le résultat (Caching) *afin de répondre aux requêtes ultérieurs directement sans transmettre la requête vers le serveur si d'autres demandes pour la même ressource sont reçus par la suite*;
- Ainsi, l'administrateur réseau peut configurer un proxy pour accélérer les réponses HTTP et économiser l bande passante ;
- Un proxy sert aussi à cacher l'adresse du client (quand le proxy effectue le travaille précédant, l'adresse IP et Port TCP dans les paquets transmis au serveur ne seront pas ceux du client, mais ceux du proxy);



Types de headers (entêtes)

IX

Le seul entête obligatoire dans une requête c'est le l'entête *Host* qui a pour valeur le nom du serveur HTTP (ou l'adresse IP de celui-ci) désigné ;

Entêtes générales <general-headers> :

- *Connection* : Le client utilise cet entête pour indiquer au serveur que la connexion doit être fermée après l renvoie de la réponse ou bien elle doit être maintenue ; Exemple : *"Connection : close" signifie que la connexion doit être fermée après l'envoie de la réponse par le serveur ;"Connection : keep-alive" signifie que la connexion doit être maintenue ouverte après l'envoie de la réponse par le serveur ;*
- *Date* : Indique la date de la génération du message (requête ou réponse) Exemple : , "*Date : Wed, 16 Sep 2018 16:43:50 GMT*"
- *Cache-Control* : Indique si les réponses / requêtes doivent être enregistrées afin de répondre aux requêtes ultérieurs localement; Exemple : "*Cache-Control : no-store*" (les potentiels proxies sur la route ne doivent pas enregistrer le contenu du message). Ce-ci est utile surtout pour les données confidentielles;
- *Trailer* : Indique l'existence d'entêtes après les données transmises car typiquement les entêtes sont transmis avant le contenu. Ainsi, toute machine qui traite le message HTTP sera averti pour séparer les entêtes des données ;
- *Transfer-Encoding* : Méthode d'encodage de tout le message HTTP ;
- *Via* : Utilisé par les machines intermédiaires pour indiquer leur interventions dans la communication ;

Entêtes requêtes <request-headers>:

- *Accept* : Types de média acceptés par le client (text, images, etc);
- *Accept-Language* : Liste les tags des langage acceptés par le client (fr , en, etc);
- *Accept-Encoding* : Utilisé souvent pour indiquer au serveur s'il est possible de transmettre les données en mode compressé. Si présent, celui-ci contiendra les algorithmes de compression supportés (gzip, deflate, etc) ;
- *If-Modified-Since :* Instruit le serveur de transmettre la ressource seulement si la date de modification est supérieure à la date indiquée par cet entête. Cet entête est utilisé par GoogleBot pour éviter de télécharger et indexer des pages non modifiée ;
- *Range* : Permet au client de demander une partie de la ressource spécifié par une intervalle d'octets (par exemple Range : bytes=1000-3000 : l'ensemble des octets de la ressource de 1000 jusqu'à 3000).
- *If-Range* : Permet d'appliquer une condition à l'entête Range. Si la condition de cet entête est valide, la partie de la ressource indiquée par Range sera renvoyée par le serveur. Sinon (la condition If-Range n'est pas valide), le serveur renvoie la ressource complète. Exemple : *If-Range: Wed, 21 Oct 2015 07:28: 00 GMT* (si la ressource n'a pas été modifiée ultérieurement à cette date, la partie spécifiée par le header Range doit être transmis par le Serveur) ; If-Range *peu être utilisée avec un hash pour vérifier si la ressource a été modifiée* ;

Entêtes du contenu <entity-headers> :

- *Content-Encoding* : Indique la méthode utilisée pour encoder la ressource contenue dans le message HTTP (compression généralement) ;
- *Content-Language* : Indique le langue de la ressource ;
- *Content-Length* : Indique la taille en octets de l'entité ;
- *Content-MD5* : Message diegest permettant la vérification de l'intégrité de la ressource (hashing);
- *Content-Range* : Permet de transmettre une partie de l'entité et de la positionner par rapport à l'entité complète. Par exemple, si la requête GET demande une partie d'un fichier de l'octet 0 jusqu'à l'octet 3000 (avec le header *Range*) d'un fichier de taille de 6000 octets, la réponse du serveur comportera un entête *Content-Range : bytes 0-3000/6000* ;
- *Content-Type* : Indique le type de la ressource (texte, image ou autre);
- *Expires* : Spécifie la date après la quelle le contenu doit être considéré comme périmé. Ainsi, si une ressource est enregistrée dans le cache et l'entête *Expires* est inférieur à la date courante, le programme client / proxy (ou navigateur) demandera la ressource à nouveau du serveur (utilisé pour le contenu dynamique) ;
- *Last-Modified* : Date de la dernière modification. La valeur de cet entête peut être utilisée pour éviter de télécharger un fichier qui n'a pas été modifié depuis le dernier téléchargement ;

Remarque : L'entête Cookie et Set-cookie

- Le serveur HTTP peut demander au navigateur client d'ajouter un entête Cookie à chacune de ces requêtes http suivantes en transmettant un message http avec l'entête *Set-cookie :[valeur-cookie-client]*. Par la suite, chaque fois que le client communique avec ce serveur, le message http contiendra un entête *Cookie :[valeur-cookie-client]*.
- La valeur du cookie est générée au début d'une manière aléatoire. Par la suite, une fois la valeur cookie est associée à un client, celle-ci permet de suivre ça navigation dans le site et de l'authentifier sans avoir besoin de recevoir son login/mot de passe dans chaque requête.

HTTPS

X

- HTTPS fonctionne exactement comme HTTP sauf qu'au début, le client et le serveur appliquent une procédure d'échange de Certificats contenant la clé de chiffrement asymétrique;
- Après la distribution de clés, la communication entre le client et serveur (dans les deux sens) sera cryptée. Par contre, les échanges entre ces deux reposent sur les mêmes entêtes/verbes HTTP ;
- HTTPS utilise le port TCP / 443 ;