

INF344 – Données du Web

HTTP

Antoine Amarilli



HTTP (HyperText Transfer Protocol), couche 7

- Le **World Wide Web** (WWW)
- **Protocole** de la navigation Web

→ Pour résumer :

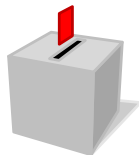
- Une machine **client**
- Un logiciel client : le **navigateur**
- Une machine **serveur**
- Un logiciel serveur : le **serveur Web**
- Un canal de communication fiable et chiffré

Présentation générale

- Standardisé par l'**Internet Engineering Task Force** (IETF) et le **World Wide Web Consortium** (W3C)
- Protocole **documenté** : RFC 2616 (114 pages, 1999, + suites)
- **Extensions** : WebSockets, nouveaux en-têtes...
- Nouvelles versions : **HTTP/2** et **HTTP/3**

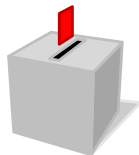
Quelle proportion des sites Web utilisent HTTP/2 aujourd'hui ?

- **A:** moins de 25%
- **B:** 25%–50%
- **C:** 50%–75%
- **D:** plus de 75%



Quelle proportion des sites Web utilisent HTTP/2 aujourd'hui ?

- **A:** moins de 25%
- **B: 25%–50%**
- **C:** 50%–75%
- **D:** plus de 75%



Versions modernes

- Nouvelle version : **HTTP/2** (originellement **SPDY** par Google)
 - Protocole **documenté** : RFC 7540 (96 pages, 2015)
 - Utilisé par **44%** des sites Web aujourd'hui ¹

1. <https://w3techs.com/technologies/details/ce-http2/all/all>, avril 2020

Versions modernes

- Nouvelle version : **HTTP/2** (originellement **SPDY** par Google)
 - Protocole **documenté** : RFC 7540 (96 pages, 2015)
 - Utilisé par **44%** des sites Web aujourd'hui ¹
- Version en développement : **HTTP/3** (novembre 2018) à partir d'un projet par Google (QUIC) pour accélérer **TCP**
 - Support expérimental dans Chrome et Firefox
 - Utilisé par **4%** des sites Web aujourd'hui (avril 2020)

1. <https://w3techs.com/technologies/details/ce-http2/all/all>, avril 2020

La requête HTTP (1.1)

- Du **client** vers le **serveur**, connexion TCP (+TLS)

`GET /wiki/Telecom_Paris HTTP/1.1`

`Host: en.wikipedia.org`

→ `http://en.wikipedia.org/wiki/Telecom_Paris`

Méthode Plusieurs possibilités :

`GET` La plus fréquente

`POST` Formulaires et effet de bord

`HEAD` Méta-infos seulement

autres `PUT`, `DELETE`...

Chemin C'est celui de l'URL

Version Ici, 1.1

En-têtes Infos supplémentaires (cf. plus tard)

Corps Passer des paramètres avec `POST`

La requête HTTP (1.1)

- Du **client** vers le **serveur**, connexion TCP (+TLS)

`GET /wiki/Telecom_Paris HTTP/1.1`

`Host: en.wikipedia.org`

→ `http://en.wikipedia.org/wiki/Telecom_Paris`

Méthode Plusieurs possibilités :

`GET` La plus fréquente

`POST` Formulaires et effet de bord

`HEAD` Méta-infos seulement

autres `PUT`, `DELETE`...

Chemin C'est celui de l'URL

Version Ici, 1.1

En-têtes Infos supplémentaires (cf. plus tard)

Corps Passer des paramètres avec `POST`

(Démon avec `netcat`)

La réponse HTTP

- Du **serveur** vers le client, dans la même connexion

HTTP/1.1 200 OK

Content-Type: text/html; charset=UTF-8

<!DOCTYPE html>

<html>

<head>

(...)

- Code d'état et explication
- En-têtes
- Réponse (e.g., corps de la page)

Codes de réponse les plus fréquents

1xx Information

2xx Succès

- 200 : OK

3xx Redirection

- 301 : permanente
- 302 : temporaire

4xx Erreur client

- 400 : erreur de syntaxe
- 401 : authentification nécessaire
- 403 : interdit
- 404 : non trouvé

5xx Erreur serveur

- 500 : erreur interne du serveur

Chemins et paramètres

- Les chemins sont **hiérarchiques** (séparateur : /)
- Conventions Unix : `https://en.wikipedia.org/./wiki/./`
- Ajout possible de paramètres **non hiérarchiques**
- **Exemple** : `https://www.google.com/search?q=telecom&ie=utf-8&oe=utf-8&client=iceweasel-a`
- Caractères spéciaux **encodés** :
`https://fr.wikipedia.org/wiki/T%C3%A9l%C3%A9com_Paris`

Table des matières

HTTP

En-têtes client

En-têtes serveur

Autres aspects

HTTP 1 vs HTTP 2

- **Plusieurs** sites Web par machine :
 - `fr.wikipedia.org` et `en.wikipedia.org`
- Le nom de domaine est **perdu** après la résolution DNS
- En-tête pour **repréciser** le domaine attendu

Host: `en.wikipedia.org`

- **Plusieurs** sites Web par machine :
 - `fr.wikipedia.org` et `en.wikipedia.org`
- Le nom de domaine est **perdu** après la résolution DNS
- En-tête pour **repréciser** le domaine attendu

Host: `en.wikipedia.org`

(Démonstration : requête à `en.wikipedia.org` avec différents Hosts.)

- Identifier le **navigateur** utilisé
- Est-ce un **ordinateur** ou un **téléphone**?
- Est-ce un **humain** ou un **robot**?
- Permet de contourner des **bugs**
- Pas de **garanties**!
- Délires historiques (tout le monde dit "Mozilla/5.0")
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:17.0)
Gecko/20130810 Firefox/17.0 Iceweasel/17.0.8

- Identifier le **navigateur** utilisé
- Est-ce un **ordinateur** ou un **téléphone** ?
- Est-ce un **humain** ou un **robot** ?
- Permet de contourner des **bugs**
- Pas de **garanties** !
- Délires historiques (tout le monde dit "Mozilla/5.0")

```
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:17.0)
          Gecko/20130810 Firefox/17.0 Iceweasel/17.0.8
```

(Démonstration : détails d'une requête de Firefox)

En-tête Accept et Accept-*

- Plusieurs **versions alternatives** d'une ressource :

Type de fichier Différents formats de page Web...

Langue Anglais, français...

Encodage Compression à la volée

- **Négociation** de contenu
- Indique plusieurs **choix** et des **priorités**
- En pratique, le type de fichier n'est pas vraiment utilisé

```
Accept: text/html,application/xhtml+xml,  
application/xml;q=0.9,*/*;q=0.8
```

```
Accept-Language: en-US,en;q=0.5
```

```
Accept-Encoding: gzip, deflate
```

- Indique de **quelle page** le navigateur vient
- Transmis après clic sur un **lien** ou soumission d'un **formulaire**
- Pas de **garanties**!
- Quid de la **vie privée**?

Referer: `https://en.wikipedia.org/wiki/Telecom_Paris`

- Demander uniquement une **certaine partie** du fichier
- Utile pour reprendre un **téléchargement interrompu** !

Range: bytes=0-42

Table des matières

HTTP

En-têtes client

En-têtes serveur

Autres aspects

HTTP 1 vs HTTP 2

- Identifie le **serveur**
- (Toujours) pas de **garanties!**

- Identifie le **serveur**
- (Toujours) pas de **garanties**!

(Démonstration : réponse à la requête Firefox.)

En-tête Content-Type et Content-Length

- Plusieurs **types de ressources** sur le Web : pages Web, mais aussi images, texte, PDF, multimédia...
 - Nécessité d'identifier le **type** (quel document, quel format?)
Internet media type, historiquement Multipurpose Internet Mail Extensions (MIME)
- Plusieurs encodages (formats de représentation du texte) :
 - Nécessité de préciser l'encodage pour certains documents
- Afficher une **barre de progression** ?
 - Préciser la **longueur** à l'avance

Content-Type: text/html; charset=UTF-8

Content-Length: 4242

Encodages

Un encodage va **numéroter** les caractères et **encoder** les numéros

ASCII 128 caractères. Pas d'accents. 7 bits

ISO-8859 128 caractères en plus pour compléter ASCII; 1 octet
Français : ISO-8859-1 alias Latin-1

Unicode Numéroter **tous** les caractères

UTF-32 Représenter les caractères Unicode; 4 octets

UTF-8 Compatible avec ASCII; longueur variable

Encodage ASCII			Encodage Latin-1		Encodage UTF-8	
	nb	représentation	nb	représentation	nb	représentation
A	65	01000001	65	01000001	65	01000001
é	-	-	233	11101001	233	11000011 10101001
ñ	-	-	-	-	324	11000101 10000011

→ En pratique : **utiliser UTF-8**

Quel est l'ordre de grandeur du nombre de caractères définis dans le standard Unicode ?

- **A:** moins de 1 000 caractères
- **B:** 10 000 caractères
- **C:** 100 000 caractères
- **D:** 1 000 000 caractères



Quel est l'ordre de grandeur du nombre de caractères définis dans le standard Unicode ?

- **A:** moins de 1 000 caractères
- **B:** 10 000 caractères
- **C: 100 000 caractères**
- **D:** 1 000 000 caractères



Table des matières

HTTP

En-têtes client

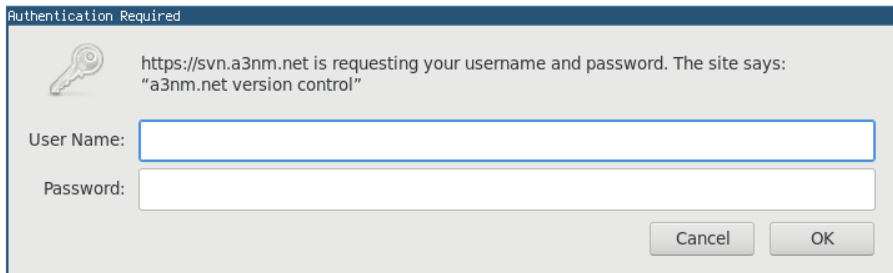
En-têtes serveur

Autres aspects


HTTP 1 vs HTTP 2

Authentication HTTP Basic et Digest

- HTTP peut **authentifier** le client avec un mot de passe



Authentication Required

 https://svn.a3nm.net is requesting your username and password. The site says: "a3nm.net version control"

User Name:

Password:

Cancel OK

- Mot de passe transmis **en clair** si pas de HTTPS
- Aussi un mode d'authentification **Digest** où le mot de passe n'est pas en clair
- Pas très **flexible** pour les sites Web
- Souvent utilisé pour des **applications** (git, svn, CalDAV, etc.)

- **Proxy** (serveur mandataire) : effectue (ou relaie) des requêtes Web pour le compte d'autrui
- Peut s'utiliser côté **client** ou côté **serveur**
- Nombreuses utilisations :
 - **Filtrer** ou **censurer** le Web (employeurs, écoles, régimes totalitaires, contrôle parental, etc.)
 - Conserver un **journal** de l'activité
 - Conserver un **cache** (cache commun à plusieurs utilisateurs)
 - **Anonymisation** (cacher la véritable origine de la requête)
 - Autres **modifications** : traduction, etc.
- Avec SSL, c'est **plus difficile** !

Content delivery networks (CDNs)

- Assurer une bonne distribution du **contenu statique**
 - e.g., JSDelivr, BootstrapCDN, Cloudflare, Google Hosted Libraries, Google Fonts
- Souvent en lien avec les **fournisseurs d'accès à Internet** (FÀI)
- Optimiser la **connexion** entre le datacenter du CDN et le fournisseur du contenu
- Proposent souvent le **filtrage des robots**, des **protections DDOS**, etc.
- Implications pour la **sécurité**, cf **subresource integrity**
- Voir aussi : **Instant Articles** par Facebook, et **Google AMP**

- **Sauvegarder** le résultat d'une requête pour la réutiliser
- Plusieurs caches possibles : proxies, ou le navigateur
- Serveur :

`Cache-Control` Indiquer si on peut conserver en cache :

`public, private, no-cache/no-store`

`Expires` Date d'**expiration**

`ETag` Identifiant de **version**

- Client :

`If-Modified-Since` Si modifié depuis une certaine **date**

`If-None-Match` Si l'identifiant de **version** a changé

- Pas de **sessions** en HTTP
- Le serveur peut demander au client de **mémoriser une valeur** :
`Set-Cookie: nom=valeur; option1; option2 :`
 - `expires` : date d'**expiration** (peut être lointaine)
 - on peut limiter la **portée** (domaine, chemin, etc.)
- Le client **fournit la valeur** avec chaque requête :
`Cookie: nom=valeur`
- Le client peut **lire** et **modifier** des cookies (pas de garanties) et les **effacer** (vie privée)

- Stocker un **identifiant de session** opaque. (Interception ?)
- Garder l'utilisateur **connecté** sur de longues périodes
- **Traquer** un utilisateur de façon unique (même sans compte)
- Risques pour la **vie privée** : pistage des utilisateurs (directive européenne sur les cookies)
- Risques pour la **sécurité** : quiconque vole un cookie peut usurper la session de son propriétaire

Table des matières

HTTP

En-têtes client

En-têtes serveur

Autres aspects

HTTP 1 vs HTTP 2

- Avec HTTP 1.1, compression **possible** si le client et le serveur la supportent
`Accept-Encoding: gzip, deflate`

- Avec HTTP 1.1, compression **possible** si le client et le serveur la supportent
`Accept-Encoding: gzip, deflate`
- Avec HTTP 2, on peut également compresser les **en-têtes**

Type de connexion

- Avec HTTP 1.0, la connexion TCP était **fermée** après chaque requête

Type de connexion

- Avec HTTP 1.0, la connexion TCP était **fermée** après chaque requête
- HTTP 1.1 : la connexion reste **ouverte** par défaut (avec temps limite)

`Connection: keep-alive`

- **Pipelining** : envoi de plusieurs requêtes, réponses **dans l'ordre**
 - Peu utilisé car **mal supporté** en pratique

Type de connexion

- Avec HTTP 1.0, la connexion TCP était **fermée** après chaque requête
- HTTP 1.1 : la connexion reste **ouverte** par défaut (avec temps limite)

`Connection: keep-alive`

- **Pipelining** : envoi de plusieurs requêtes, réponses **dans l'ordre**
 - Peu utilisé car **mal supporté** en pratique
- Avec HTTP 2, **multiplexing** : envoyer plusieurs requêtes et réponses dans un **ordre arbitraire**
- Avec HTTP 2, le serveur peut aussi envoyer **de sa propre initiative** des ressources au client

- Matériel de cours inspiré de notes par Pierre Senellart et Georges Gouriten
- Merci à Pierre Senellart pour sa relecture