Le protocole HTTP et le standard MIME

Ce petit tutoriel va vous donner quelques notions sur ce qui se passe sous le capot: comment le navigateur et le serveur échangent des données.

Un peu de culture: le modèle OSI

Les communications réseaux fonctionnent en couches: c'est le **modèle OSI** (pour *Open Systems Interconnection*). Ce modèle décrit comment le réseau fonctionne, du plus concret au plus abstrait: chaque couche fournit une abstraction supplémentaire.

Niveau 1 : La couche physique

Il s'agit de la couche la plus basse: comment faire pour représenter des données à partir de signaux physiques ? Les protocoles de cette couche proposent des solutions pour interpréter des signaux électrique en bits (0/1, vrai/faux).

On trouve par exemple les protocoles physiques de communication filaires: DSL, USB, les protocoles pour les fibres optiques, le protocole des modem téléphoniques, et bien sûr la couche ethernet filaire standard avec les cordons RJ45. Mais on trouve aussi la couche physique de Bluetooth, les réseaux wifi 802.11, ...

Niveau 2 : La couche liaison

Cette couche part du principe qu'elle a accès à la notion de bits: elle va définir la notion de trame, c'est à dire la façon d'agencer les bits pour communiquer. Typiquement, cette couche va attribuer une adresse à chaque interface connectée, va définir comment les données sont encapsulée (les trames), et va définir la façon dont les trames circulent entre les interfaces.

C'est sur cette couche que l'on va trouver:

- la notion d'"ethernet" telle qu'on la connait pour internet. C'est à se niveau que l'adresse MAC des interfaces fait sens.
- mais aussi les "anneau à jetons" (*token ring*): une trame appelée jeton circule dans une seule direction autour d'un anneau. L'avantage est l'absence de collision.
- ou encore ARINC 429: une norme dans l'avionique.

Niveau 3 : La couche réseau

C'est la dernière couche des couches dites matérielles. Cette couche détermine le parcours des données et l'adressage logique.

Pour l'internet, c'est la couche dite IP: on s'abstrait de l'adresse MAC, et la notion de réseau devient une notion logique, indépendante de la couche physique.

Niveau 4 : La couche transport

Première couche dite "applicative", elle décrit les communications entre processus. Cette couche est la dernière à gèrer et corriger les erreurs : elle garantit une communication sans corruption et idéalement sans perte.
C'est la couche des protocoles UDP (*User Datagram Protocol*) et TCP (*Transmission Control Protocol*). TCP permet de garantir une connexion fiable entre deux processus qui peuvent ensuite échanger des données. C'est le protocole utilisé classiquement dans tous les protocoles de plus haut niveau des couches applicatives du dessus, en particulier... HTTP.

Niveau 5, 6 et 7: Les autres couches applicatives.

Ces trois couches, respectivement session, présentation et application, sont les couches spécialisées pour l'échange de données. Dans le contexte IP (donc de l'internet), la distinction n'est pas très nette: de façon générale, c'est à ce niveau que se passe les protocoles utilisés par les applications:

- DHCP
- HTTP pour le web
- IMAP et POP pour le mail
- FTP pour le transfer de fichier
- SSH pour la communication crypté
- VoIP pour la téléphonie sur internet…

Le protocole HTTP

Le protocole HTTP (*Hypertext Transfer Protocol*) est un protocole de communication de type requête / réponse synchrone. Le protocole est textuel: les messages échangés utilisent des caractères encodés sur 8 bits. Une *session HTTP* consiste en une série d'émissions alternées entre un client et un serveur. Chaque émission consiste en un chapeau, un entête et un corps de message.

Les requêtes

La requête provient du client et s'adresse au serveur.

Le chapeau

Il contient la **commande HTTP**: ce que l'on demande au serveur. Typiquement:

- GET: demande de resource. En principe, cette requête est sans effet au niveau du serveur.
- POST: transmission de données, typiquement avec la création ou la modification de données au niveau du serveur (par exemple, mise à jour d'une base de donnée)

Il y a quelques autres commandes, mais pour les besoin de ce cours ces deux commandes suffisent.

Le GET et le POST prennent deux arguments:

- L'adresse de la resource (à partir de la racine du serveur)
- La version d'HTTP préférée

Par exemple:

```
1  GET /index.html HTTP/1.1
```

va demander la resource `index.html` qui se situe à la racine du site, et demander la version 1.1 du protocole.

L'entête

L'entête est une série de paires (nom d'attribut, valeur). Pour une requête typique, on trouve les attributs

- `Host`: le domaine du site internet concerné par la requête
- `Referer`: L'url du document d'où on vient
- `User-Agent`: chaine de caractère indiquant le nom du navigateur (ou de façon plus général du logiciel qui fait la requête).
- `Cookie`: Les cookies éventuels que le site internet avait alloués

Le corps du message

Il est séparé de l'entête par une ligne vide.

Pour un GET, le corps du message est vide. Pour un POST, le corps du message contient le contenu des données que l'on envoie au serveur. Par exemple, le contenu d'un formulaire.

Les réponses

Une réponse provient du serveur, et correspond à une requête.

Le chapeau

Il consiste en un code de statut HTTP. Il est sous la forme

```
HTTP/version code-réponse texte-réponse
```

Un sous-ensemble des codes et textes des réponses:

- 200 OK
- 301 MOVED PERMANENTLY
- 308 PERMANENT REDIRECT
- 400 BAD REQUEST
- 401 UNAUTHORIZED
- 403 FORBIDDEN
- 404 NOT FOUND
- 500 INTERNAL SERVER ERROR

L'entête

Comme pour les requêtes, l'entête contient des méta-données. Par exemple:

- `Date`: la date de création
- `Server`: le nom du programme qui fait tourner le serveur
- `Last-Modified`: la date de dernière modification
- `Content-Type`: le type MIME (voir plus bas) et l'encodage
- `Content-Length`: la taille du corps du message
- `Set-Cookie`: pour allouer des cookies

Le corps du message

Contient les données demandées par la requête. L'encodage et le type des données sont donnés dans l'entête.

Exemple

Par exemple, la requête suivante sur le port 80 du serveur domain.com:

```
GET /index.html HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:43.0) Gecko/20100101 Firefox/43.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: keep-alive

```

Donne comme réponse

?

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: max-age=604800
Content-Type: text/html
Date: Wed, 17 Feb 2016 14:50:13 GMT
Etag: "359670651"
Expires: Wed, 24 Feb 2016 14:50:13 GMT
Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
Server: ECS (iad/18F0)
Vary: Accept-Encoding
X-Cache: HIT
x-ec-custom-error: 1
Content-Length: 1270

<!doctype html>
<html>
<head>
    <title>Example Domain</title>
    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 50px;
        background-color: #fff;
        border-radius: 1em;
    }
    a:link, a:visited {
        color: #38488f;
        text-decoration: none;
    }
    @media (max-width: 700px) {
        body {
            background-color: #fff;
        }
        div {
            width: auto;
            margin: 0 auto;
            border-radius: 0;
            padding: 1em;
        }
    }
    </style>
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is established to be used for illustrative examples in documents. You
    domain in examples without prior coordination or asking for permission.</p>
    <p><a href="http://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>
```

Le standard MIME

Lorsqu'on reçoit une réponse HTTP, il n'y a *à priori* aucun moyen de savoir quel est le type de donnée: est-ce du texte brut, du XML, un fichier compressé, une vidéo, ... Et dans le cas d'un texte, quel est l'encodage des caractères. On reçoit des octets en pagaille, et il faut pouvoir en faire du sens. C'est le but du pragma `Content-Type` dans l'entête de la réponse: dire comment interpréter le corps du message.
Le standard MIME (pour *Multipurpose Internet Mail Extension*), à l'origine développé pour les mails (qui ont le même type de problème avec les fichiers attachés), est aussi utilisé pour HTTP.

Le format

De façon général, le type MIME et l'encodage du document est donné par
?
```
1   Content-Type: type/sous-type; charset=jeuDeCaractères
```
L'encodage correspond à la façon d'associer des octets à des caractères... Et il y a beaucoup de façons de faire. Il faut donc le dire,

Quelques exemples de types MIME.

text

- text/plain : pour du texte brut
- text/html : pour de l'HTML
- text/xml : pour du XML
- text/javascript : pour... du javascript
- text/css
- ...

image

Correspond en général au format de l'image.

- image/png
- image/jpeg
- image/gif
- image/svg+xml
- ...

video et audio

Comme pour les images, correspond en général au format.

- video/mpeg
- video/mp4
- video/quicktime
- audio/x-wav
- audio/mp3
- ...

application

Ce type sert pour les fichiers multi-usages.

- application/pdf
- application/zip
- ...

Remarque

Les types MIME sont essentiellement une convention et un peu redondant. Par exemple, on trouve `text/javascript` et `application/javascript`. On trouve `audio/wav` et `audio/x-wav`. Il faut donc une bonne entente entre l'émetteur qui choisit un type MIME et le récepteur qui va le lire...

Il faut néanmoins noter que pour les types "standards", tous le monde est à peu près d'accord.