L’administration d’un serveur Apache sous Linux

Module 6 – Le protocole HTTP

Objectifs

- Découvrir le protocole HTTP

Introduction

- Le protocole **HTTP** est un protocole de la couche Application du modèle OSI
- On utilise le protocole TCP comme couche Transport
- Le serveur HTTP répondra alors par défaut sur le **port 80**
- On rajoute la surcouche SSL/TLS pour sécuriser le transport ; le port pour le **HTTPS** est le **port 443**

Principe de fonctionnement

- Le protocole HTTP est un protocole stateless (sans état), c’est-à-dire que chaque élément constituant une page web est récupéré indépendamment

  - Connexion du **client HTTP** au serveur
  - Envoi d'une requête spécifiant la méthode suivie de l'objet cible
  - Réponse du **serveur HTTP**
  - Le serveur ferme la connexion pour signaler la fin de la réponse

Le protocole HTTP

Principe de fonctionnement

- Pour **chaque élément d'une page** (HTML, CSS, PNG…), cette procédure est reproduite
- Ceci explique en partie la lenteur du protocole HTTP

HTTP/1.1

- La version HTTP/1.1 introduit les connexions persistantes
  Cela permet de ne pas couper la connexion après l'envoi de chaque élément

- On parlera de **keepalive**

- Le chargement est ainsi accéléré et le réseau moins occupé

- La valeur du keepalive est paramétrable

Le protocole SPDY

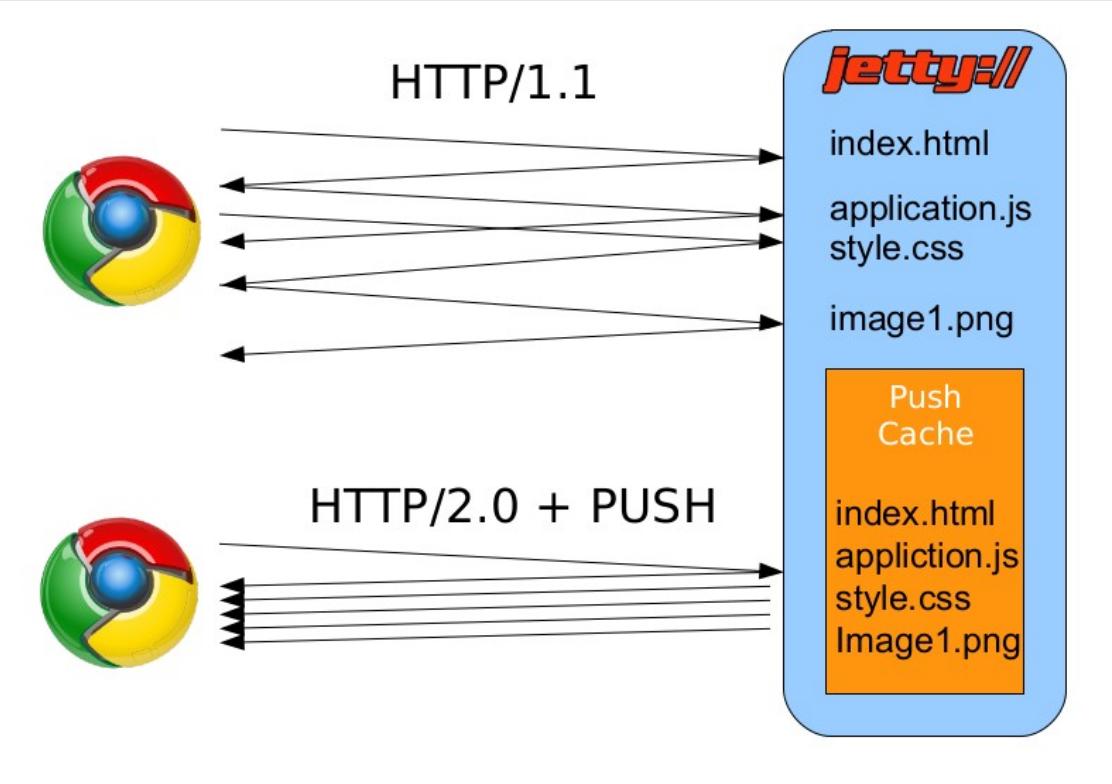
- En 2009, Google développe le protocole **SPDY**
- Celui-ci permet notamment d’utiliser une seule connexion pour un ensemble de requêtes
- Il peut envoyer les requêtes et les réponses en parallèle, ce que ne fait pas le HTTP/1.1 (une seule requête/réponse à la fois)

Le protocole HTTP

HTTP/2

- En 2015, l'IETF introduit le HTTP/2
  - Basé sur le **module SPDY**
  - Évolution de la compression des données transférées
  - Généralisation du **HTTPS**
  - Requiert des algorithmes de compression forts
  - Définition de priorités sur certaines ressources

Le protocole HTTP
HTTP/2



Source : auteur Simone Bordet

Les méthodes

- Les méthodes permettent de spécifier le type d'action à réaliser

- Les plus utilisées :
  - **GET** : demande la récupération de l'objet
  - **POST** : soumission de données (formulaires)
  - **HEAD** : demande d'informations sur un objet

- Voir la liste complète des méthodes : https://fr.wikipedia.org/wiki/Hypertext_Transfer_Protocol

La commande curl

- La commande **`curl`** permet d'afficher en ligne de commande les résultats des requêtes :
  - Afficher seulement les en-têtes :

```
# curl --head http://192.168.10.19
```

  - Afficher le contenu d'une page :

```
# curl http://192.168.10.19
```

Les en-têtes

- Les en-têtes définissent les informations de l'objet
- Chaque ligne définit un en-tête (*header*)
- Chaque en-tête est défini par son nom et sa valeur

```
Date: Wed, 02 Nov 2016 13:52:23 GMT
Server: Apache/2.4.10 (Debian)
Last-Modified: Wed, 02 Nov 2016 13:51:59 GMT
```

Les codes de statut

- Lors d'une requête, la première réponse du serveur est un code de statut

- Il existe cinq niveaux de code :

  - **Informations** : 1xx (exemple, 101 : changement de protocole)
  - **Succès** : 2xx (exemple, 200 : succès de la requête)
  - **Redirection** : 3xx (exemple, 301 redirection permanente)
  - **Erreur client** : 4xx (exemple, 404 : page non trouvée)
  - **Erreur serveur** : 5xx (exemple, 500 : erreur serveur)

Conclusion

- Vous comprenez le fonctionnement du protocole HTTP