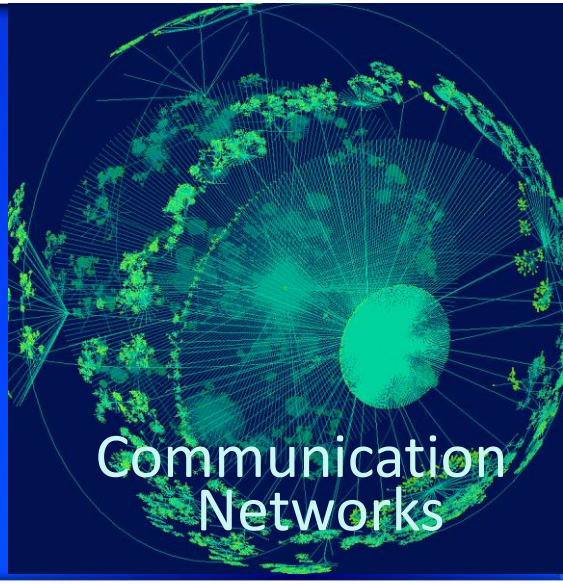
Chapitre 2 :

Couche Application



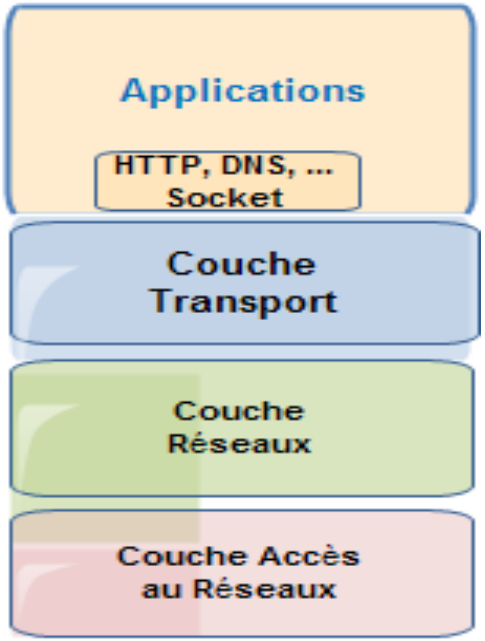
Module : Réseaux de Communication

Deuxième Année Licence Informatique

Mr A. Dekhinet
Université de BATNA 2
Département d'Informatique

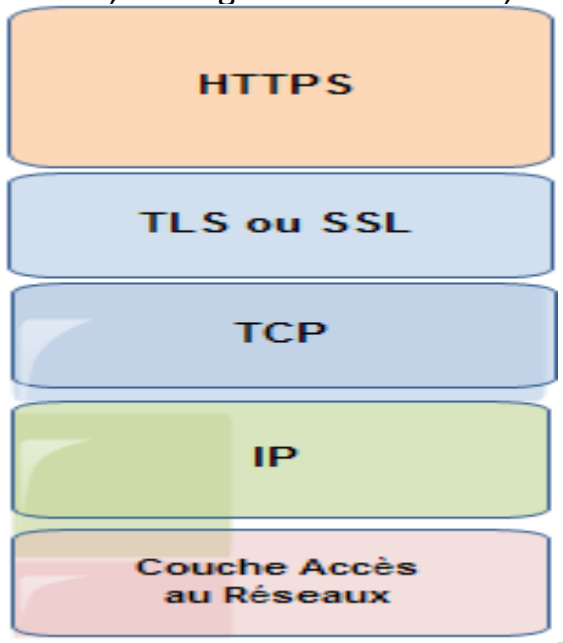
La couche Application

- La couche la plus importante et la plus visible dans les réseaux
- Les applications réseaux résident dans cette couche
- Les utilisateurs finaux interagissent avec le monde de communication (Couches inferieures) via cette couche
- Permet de mettre en place des applications distribuées (Applications réseau, Services réseaux) , en utilisant les services de transport
- Les protocoles de la couche application font, en général, partie des applications. Ils ne possèdent pas de GUI, tels que HTTP, SMTP, FTP, DNS, …
- Le protocole de la couche application, à l'instar d'autres protocoles, définit :
  - le type de message échangé : Requête ou Réponse
  - La syntaxe du message : Les champs
  - La sémantique du message : La signification de chaque champs
  - Les règles du protocole : Quand et comment les entités paires envoient et répondent au messages
- Exemples d'applications réseau :
  E-mail, Web, Messagerie instantanée, Remote login, Partage de fichier P2P,
  Jeux réseau , Streaming stored video (Youtube), VoIP (Skype),
  Real-time vidéo conférence, IPTV, Réseaux sociaux, …



Besoins des applications : Services

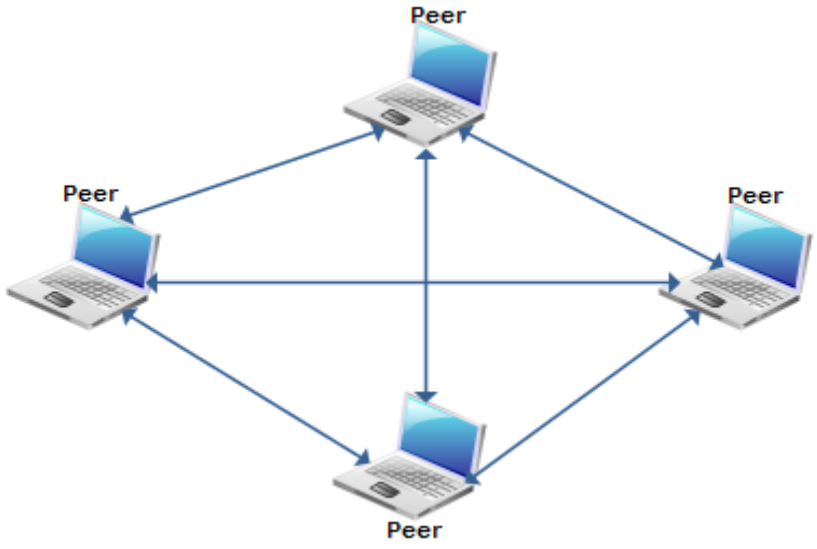
- Le choix des services , et par conséquent le protocole adéquat, dépend de la nature de l'application. Les services fournis aux applications sont divers :
  - ***Transfert fiable de données*** : Certaines applications nécessitent une fiabilité maximale (Transfert de fichier/ Download , Transaction BD, Document Web, mail, …). D'autres peuvent tolérer des pertes (Audio, Vidéo,…)
  - ***Timing*** : Certaines application sont sensibles au timing (Retard, …). Elles exigent une qualité de service (QOS) et ne tolèrent pas la gigue (variation de la latence), comme la Voix, les Jeux Interactifs, Real Time Audio/Vidéo, …
  - ***Bande passante*** : Certaines applications nécessitent un seuil minimal de la bande passante pour être effectives (Audio: 5Kbps-1Mbps, Vidéo: 10Kbps-5Mbps, …)
  - ***Sécurité*** : Certaine applications exigent des services et des mécanismes de sécurité (Cryptographie, Hachage, …). Et ce pour assurer la confidentialité des données, l'intégreté des données, l'authenticité, … , telles que les transactions bancaires
- Exemple : Cas du TCP/IP
  HTTP, FTP, SMTP, Telnet : TCP
  DNS, TFTP, RTP : UDP
  HTTPS : TLS, SSL (Secure Socket Layer)

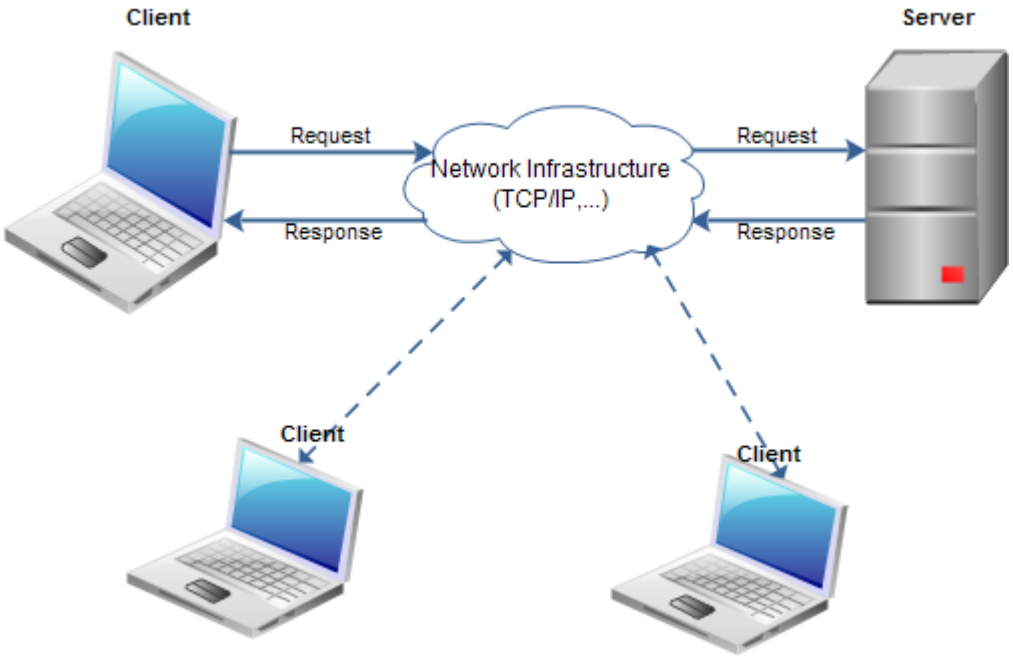


Modèles d'Applications

Paradigmes ou modèles d'applications distribuées

- Deux modèles de base sont à distinguer : Client /Serveur (CS) et Peer-To-Peer (P2P)
- Client/Serveur
  - Les processus communicant sont asymétriques.
  - Le processus client initie la communication et demande un service auprès du processus serveur
  - Le serveur, en état d'écoute (Listening), répond en fournissant le service demandé
  - Exemple : HTTP, FTP, SMTP, DNS, ….
- Peer-To-Peer (pair à pair)
  - Les processus communicants sont symétriques.
  - Le peer est client et serveur en même temps
  - Les processus communicants peuvent réaliser des taches et des fonctions similaires
  - Exemple : Skype, IRC, Torrent, ….





Un protocole d'Application : HTTP

Un protocole de la couche application : HTTP (HyperText Transfert Protocol)

- Versions : HTTP/1.1 RFC 2616, HTTP/1.0 RFC 1945, Dernière version HTTP/2.0 RFC 7540
- HTTP transporte divers types de données (Objets Web) : Texte, Image, Vidéo, Audio, …
- MIME (Multi-purpose Internet Mail Extensions) : Types de données transportées dans Internet
  Format : Type/Sous-Type,
  Exemple : image/gif, image/jpeg, text/html, application/pdf
- Concepts de HTTP
  - Client : Navigateur (Browser). Exemple: Internet Explorer, FireFox
  - HTTP Server : Processus (Daemon) Serveur. Exemple: Apache, IIS
  - HTML (HyperText Markup Language) : langage de description d'écriture des pages Web
  - Page Web : Consiste en un ensemble d'objets
  - Objet : Texte, Image, Vidéo, Fichier HTML, Fichier Audio, Applet Java, …
  - Un objet est adressable par une URL (Uniform Resource Locator)
- Dans l'URL la partie machine sera traduite en adresse, par un serveur de noms DNS, pour être insérée dans des PDUs des couches inferieures

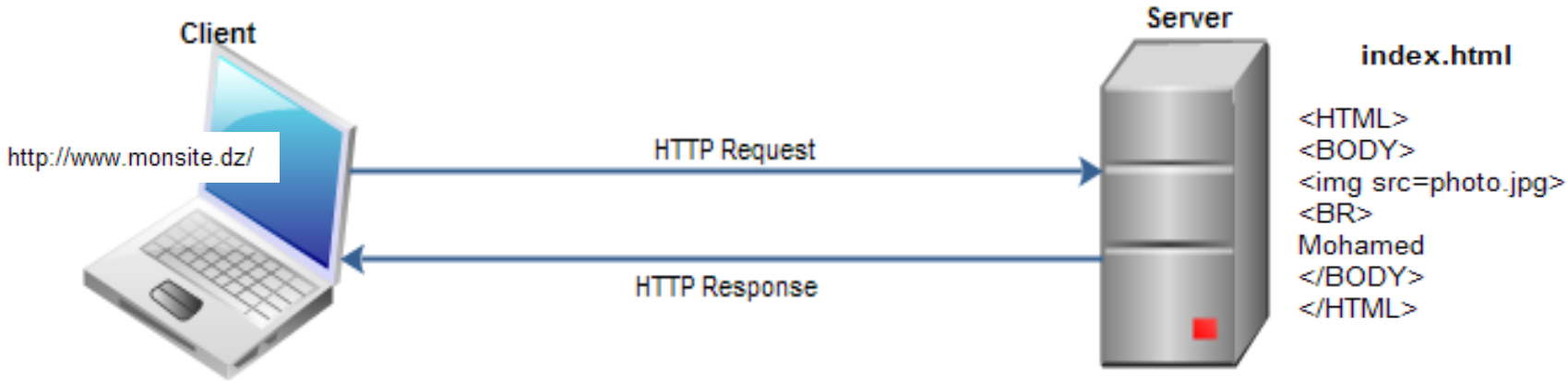
Ligne de Commande :

> telnet www.univ-batna.dz 8080

GET /dept/inf.gif HTTP/1.1

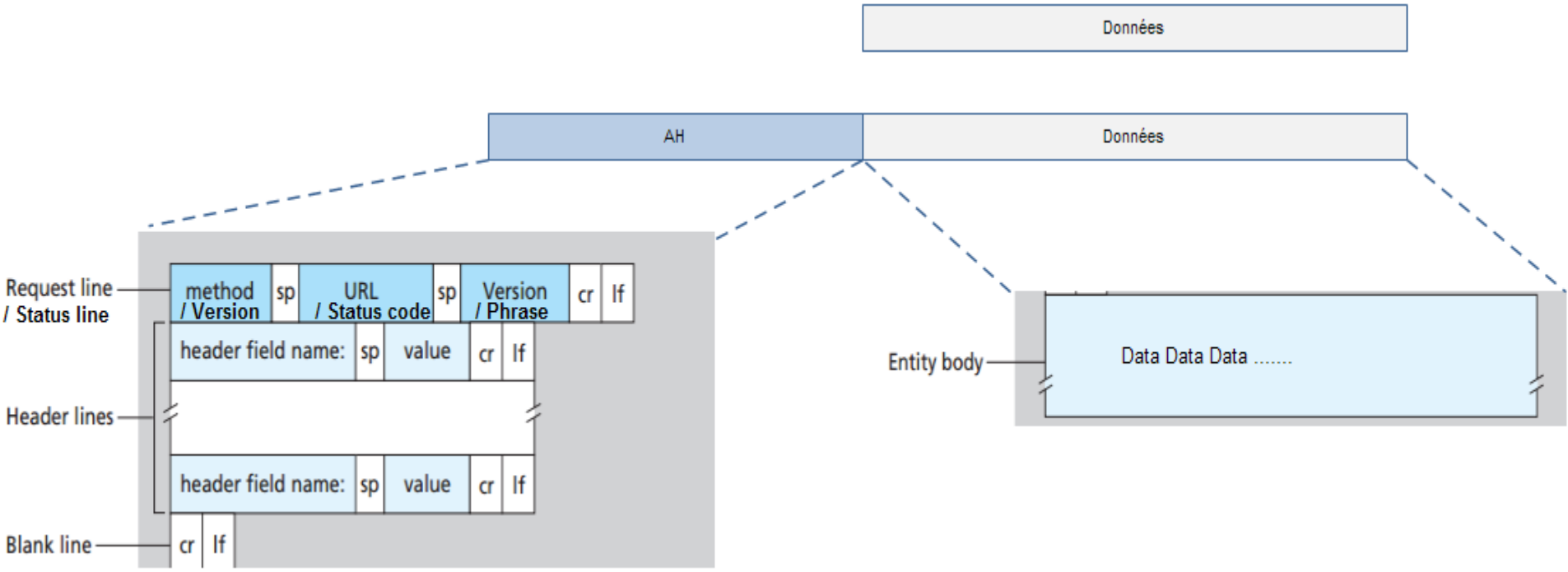
Un protocole d'Application : HTTP

- HTTP utilise TCP (TSAP= Numéro de port standard 80)
  Mode connecté : Connexion TCP, Transfert de données (Messages HTTP), Déconnexion TCP
- HTTP : Client/Serveur
- Les processus (Client ou Serveur) sont identifiés par un couple : Adresse IP et Numéro de port
- Pourquoi le PID OS n'est pas utilisé pour identifier le processus ?
- Stateless : Le serveur ne maintient aucune information (Historique des requêtes) du client
- RTT (Round Trip Time) : Temps de transfert (Aller et Retour) de PDU de petite taille. Il inclut le temps de propagation, le temps de traitement et le temps d'attente dans les systèmes intermédiaires
- PLT (Page Load Time) : Dépend de la structure et du contenu de la page web (taille des objets, gzip, …) , le RTT, la bande passante, HTTP (TCP), …
- Amélioration des performances de HTTP : Réduction du PLT
  - Persistance de la connexion
  - Web cache : Local et Proxy
  - Cookies



PDU du protocole HTTP

- PDU = Message HTTP
- Message HTTP : Requête ou Réponse
- Les données applicatives (Texte html, image, …) sont encapsulées par un entête (AH) pour former le message HTTP
- L' entête contient plusieurs champs, repartis sur un ensemble de lignes
- La sémantique des champs dépend du type du message HTTP : Requête ou Réponse
  Requête : Première ligne = Request line, Entity body= peut être vide (Dépend de la méthode)
  Réponse : Première ligne = Status line, Entity body= données (Data)



PDU du protocole HTTP

Requête HTTP typique

GET /index.html HTTP/1.1 — /Method = GET, POST, PUT, HEAD, DELETE, TRACE, …
Host: www.univ-batna2.dz
Connection: close — **/ Close : Connection non persistante**
User-agent: Mozilla/5.0
Accept-language: fr
CR/LF

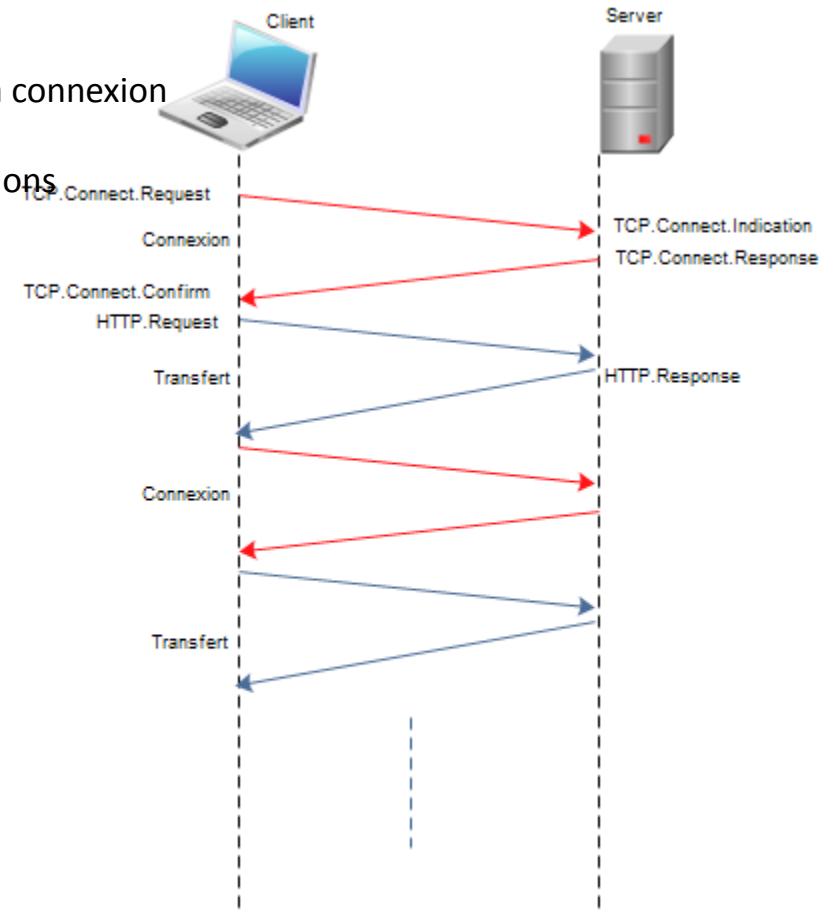
Réponse HTTP typique

HTTP/1.1 200 OK
Date: Sun, 20 Jan 2015 20:09:20 GMT
Server: Apache/2.0.52 (CentOS)
Last-Modified: Tue, 30 Oct 2014 17:00:02 GMT
Content-Length: 200
Connection: Keep-Alive — **/ Keep-Alive : Connection persistante**
Content-Type: text/html; charset=ISO-8859-1
CR/LF
……..<html><body> Mohamed </body></html>…. — **/Entity body= Data**

Status Codes :
- 200 OK
- 400 Bad Request
- 404 Not Found
- 505 HTTP Version Not Supported
- Etc …

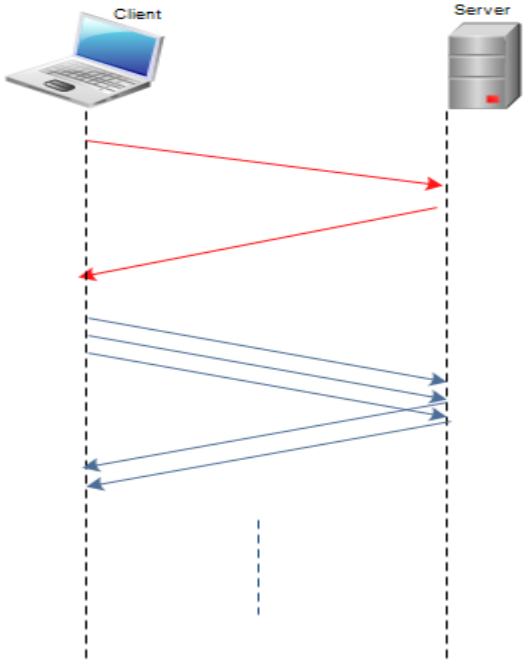
- **Persistance de la connexion**
  - La connexion HTTP peut être persistante ou non persistante
  - HTTP/1.0 : Non persistante
  - HTTP/1.1 : Par défaut persistante
  - HTTP/2.0 : Persisatnt

- **HTTP Non persistant**
  - Etablir connexion TCP, Transférer un objet, Fermeture de la connexion
  - Temps total de transfert d'un objet Web = 2RTT + T
  - Le transfert de plusieurs objets nécessite plusieurs connexions 2RTT par objet
  - Les navigateurs ouvrent , généralement, des connexions parallèles
  - **Exemple** : Un fichier index.html qui fait référence à 8 objets JPEG. Le processus de transfert s'exécute 9 fois .

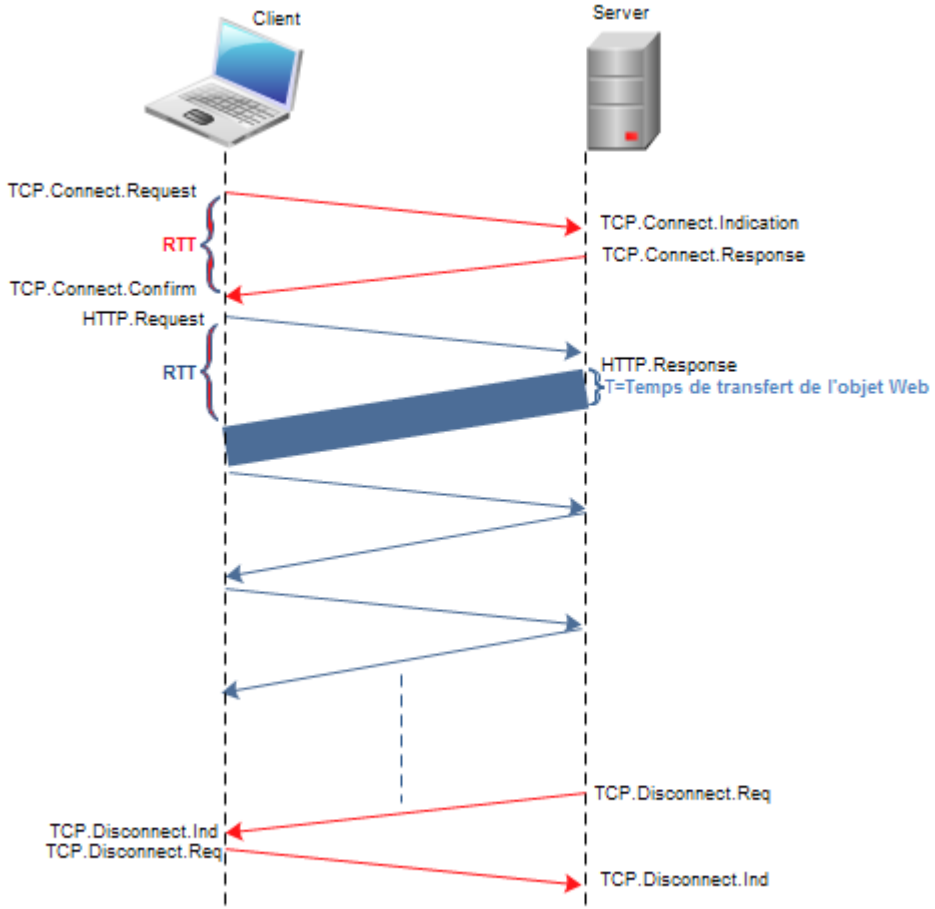


HTTP : Connexion Non Persistante

- **HTTP Persistant**
  - Etablir la connexion TCP, Envoyer plusieurs objets, Fermeture de la connexion
  - Plusieurs objets peuvent être transférés par connexion TCP
  - Réduction de la surcharge du serveur. Le serveur maintient la connexion ouverte après le transfert d'un objet
  - Permet le pipeline : Persistant avec Pipeline et Persistant sans Pipeline
  - Persistant sans Pipeline : 1 RTT pour chaque objet
  - Persistant avec pipeline :
    - Par défaut du HTTP/1.1
    - Le client émet plusieurs requêtes
    - Quand la taille des objets référencés est petite, un seule RTT leur suffira (Plusieurs PDU par RTT)



HTTP : Connexion Persistante avec Pipeline



HTTP : Connexion Persistante

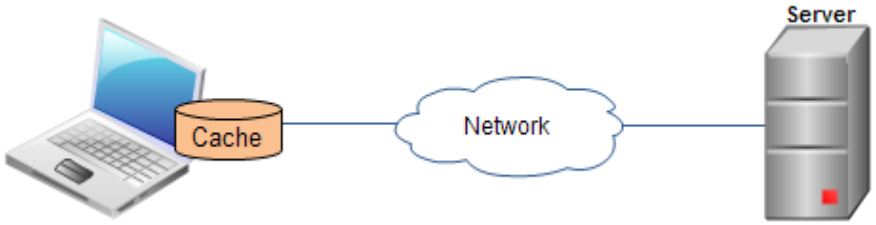
Amélioration des Performances de HTTP : Web cache

- **Web cache**
  - Réduire le PLT par réutilisation, en conservant des copies des objets déjà transférés (très) proche du client
  - Le cache peut être local (Browser) ou système dédié (Proxy)

- **Browser cache**
  - Maintient un cache local
  - L'algorithme suivant illustre le principe du cache

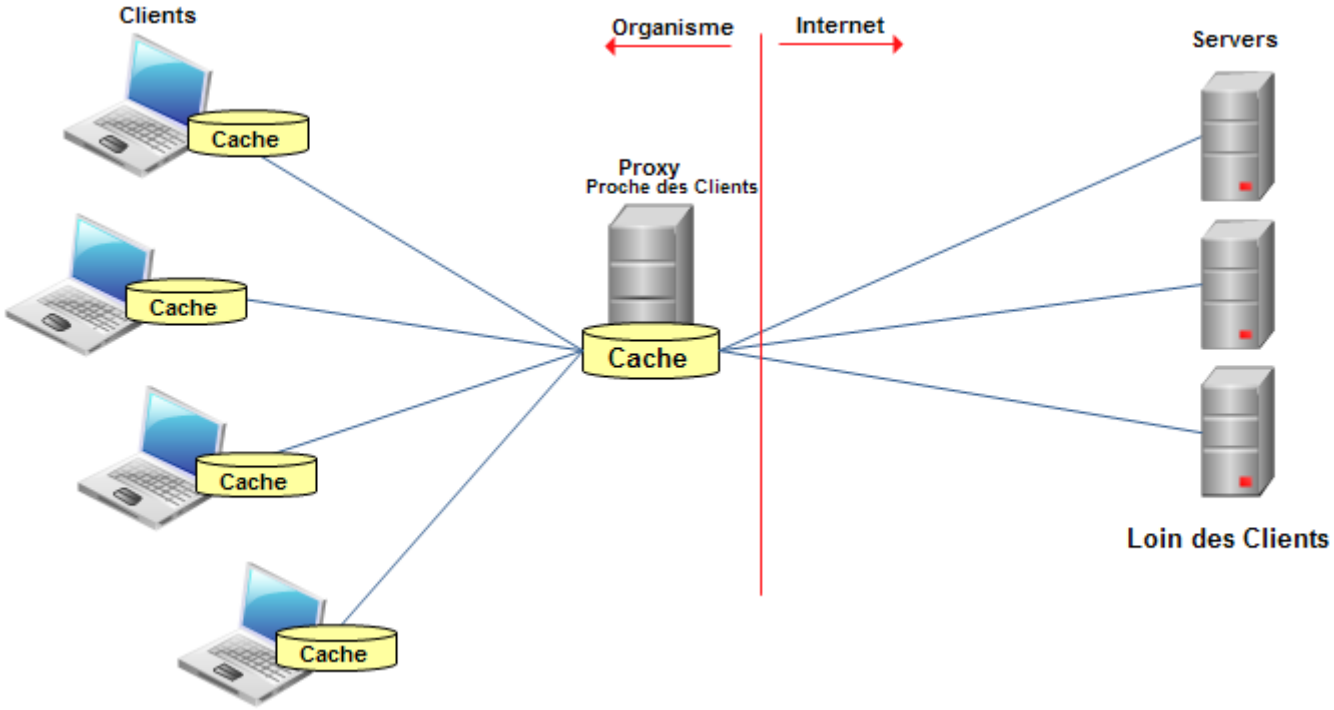
Given : A URL for an item on a web page
Obtain : A copy of the item
Method :

```
if (item is not in the local cache) {
    Issue GET request and place a copy in the cache;
} else {
    Issue HEAD request to the server;
if (cached item is up-to-date : ) {
    use cached item;
} else {
   Issue GET request and place a copy in the cache;
}
}  /End if
```



  - La validité de la copie locale dépend des informations d'expiration : Last-Modified, Etag

- **Web proxy**
  - Mise en place d'un intermédiaire (proxy) entre un pool de clients et les serveurs externes
  - Le proxy peut être une machine dédiée (Appliance), tel que BlueCoat, ou un software, tel que SQUID.
  - En plus du cache, la politique de sécurité (Les règles) d'un organisme peuvent être définies et assurées par le proxy
  - Utilisation des informations de l'entête :
    - GET conditionnel
    - Le champ if-modified-since
    - Le statut 304 Not Modified
    - …



- **Conditional GET**
  - Les objets à jours ne sont pas transférés
  - Gain en bande passante
  - Utilisation du champ : if-modified-since
  - L'exemple suivant illustre le principe du GET conditionnel

Client 1:

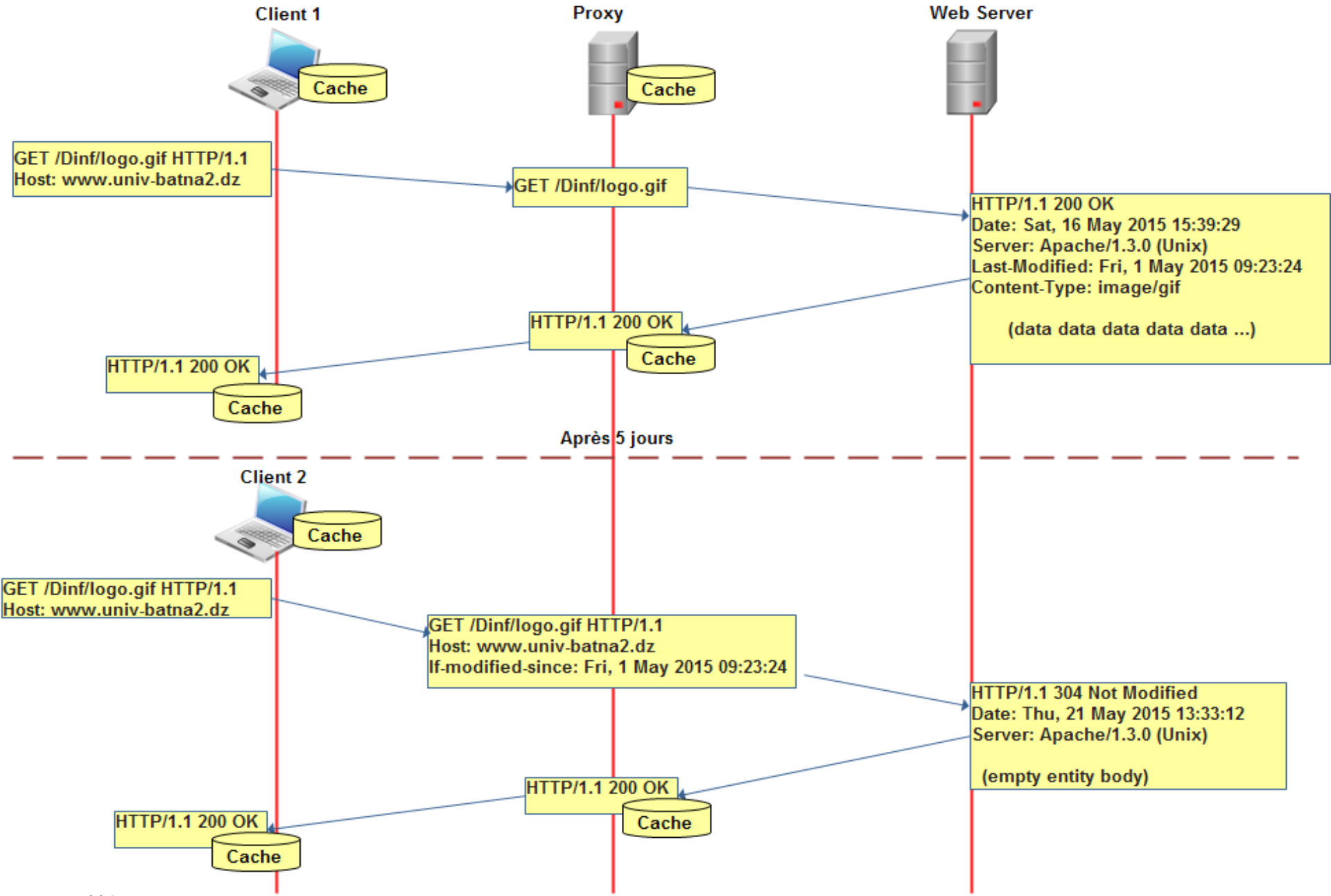
- Le client 1, via son navigateur, envoi une requête HTTP au proxy, demandant l'objet **logo.gif** (image)
- Le proxy, après consultation de son cache, redirige la requête vers le serveur Web
- Le serveur Web envoi une réponse HTTP contenant l'image demandée
- Le proxy cache l'image et l'envoi au client

Après 5 jours

Client 2:

- Un client 2, sur une autre machine, envoi une requête HTTP au proxy, demandant le même objet **logo.gif**
- Le proxy consulte son cache, et trouve l'objet, mais il doit s'assurer de sa validité (Expiration). Il envoi une requête HTTP au serveur Web, pour voir si l'objet en question a été mis à jour depuis la dernière date de mise en cache
- Le serveur Web envoi une réponse HTTP, indiquant la validité de l'objet
- Le proxy envoi une réponse HTTP au client contenant l'objet (l'image) demandée

Amélioration des Performances de HTTP : Web cache



Cookies

- HTTP est un protocole Stateless : Le serveur HTTP ne garde aucune trace (états) du client
- Cookies : Mécanisme HTTP de gestion et de maintient des informations d'états du client
- Cookies : Petits fragments d'informations (Chaîne de caractères), générés par le serveur, et stockés dans le client
- Exemples d'informations : Autorisation, Authentification, Carte de Crédit, Préférences client, Navigateur, OS, …
- La technologie de cookies est fondée sur quatre composants :
  - Ligne d'entête dans la réponse HTTP (*Set-cookie*) :
    Set-Cookie: name=*value*[; expires=*date*][; domain=*domain*][; path=*path*][; secure] [; HttpOnly]
  - Ligne d'entête dans la requête HTTP (*Cookie*)
    Cookie: name1=value1; name2=value2; ….
  - Un fichier cookie stocké dans le client et géré par le navigateur
  - Une base de données potentielle d'arrière plan, stockée et maintenue par le serveur

Amélioration des Performances de HTTP : Cookies