HTTP

IUT1 dpt SRC L'Isle d'Abeau
Jean-françois Berdjugin

HTTP

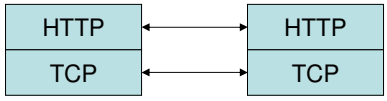
- Introduction et architecture
- Messages
- Authentification
- Conclusion

HTTP

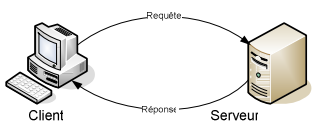
Introduction et architecture

Hypertext Transfert Protocol

- URI (Uniform Resource Identifier) = URL (Uniform Resource Locator) + URN (*Uniform Resource Name*)
- URL ⇔Adresse Postale (adresse)
- URN ⇔No INSEE (identifiant)
- Exemple d'URL: http://guest:secret@www.ietf.org:80/html.charters/ww-dir.html?ses=1#Application_Area ⇔protocol://username:password@host:port/path/file?query#fragment



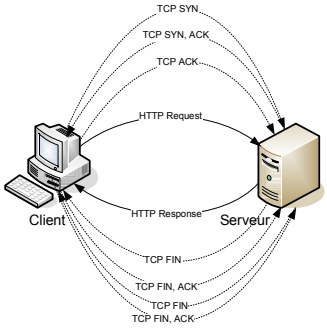
Modèle en couche



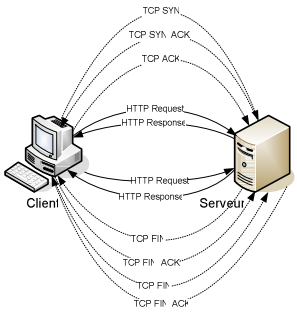
Client/Serveur

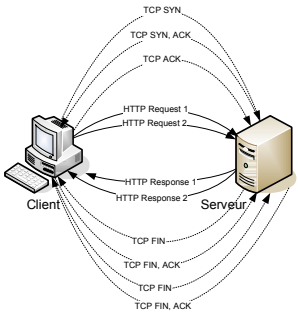
HTTP 1.0 et TCP

- HTTP 1.0 : pour chaque ressource, une ouverture de connexion, une requête, une réponse, une libération de connexion, …
- Pb : consommation de la bande passante
- Avantage : limite le « travail » des serveurs



HTTP 1.1 et TCP





Deux solutions :
- La persistance de la connexion
- Le Pipelining

=>
- Meilleure utilisation de la bande passante
- Plus de travail pour les serveurs

Structure d'une requête et d'une réponse

Request-Line
Message Headers (optional) General Headers Request Headers Entity Headers
Blank Line
Message Body (optional)

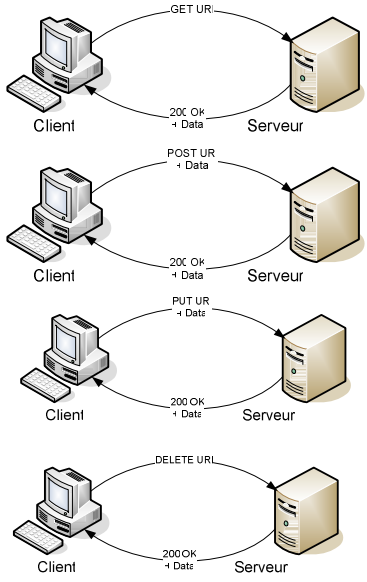
Requête HTTP

Status-Line
Message Headers (optional) General Headers Request Headers Entity Headers
Blank Line
Message Body (optional)

Réponse HTTP

Opérations utilisateur

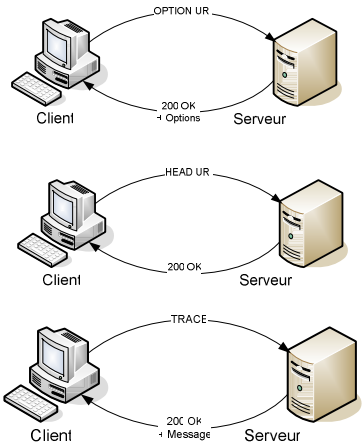
- GET : Récupérer une ressource
- POST : Envoi d'information à un objet (URI : script, programme)
- PUT : Envoi de données (URI : path + file)
- DELETE : Suppression d'objet



Autres opérations

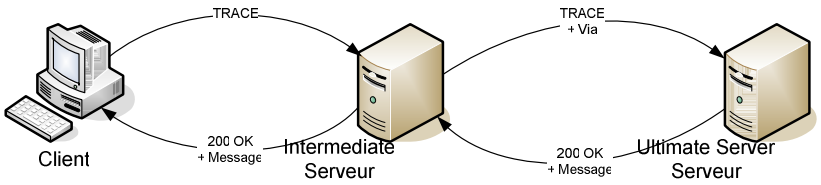
- OPTIONS : découverte des capacités du serveur (URI ou *)
- HEAD : identique à GET mais sans envoi de données
- TRACE : traçage du chemin réseau (réexpédition de ce qui a été reçu) ?

=> Coopération de serveurs



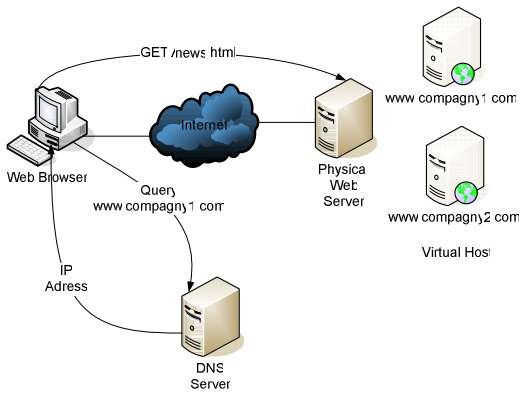
Coopération de serveurs

- Virtual Host, Redirection
- Proxy, Gateways et Tunnels



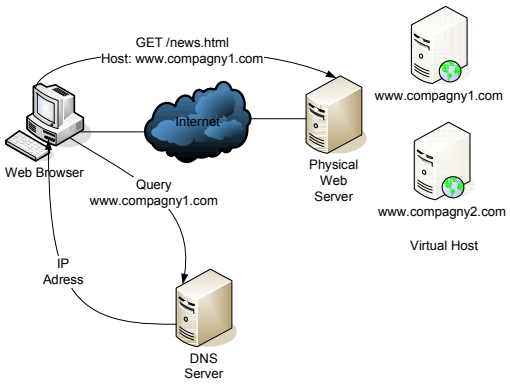
Virtual Host

Problème un provider héberge sur la même machine plusieurs sites Web => comment trouver la ressource



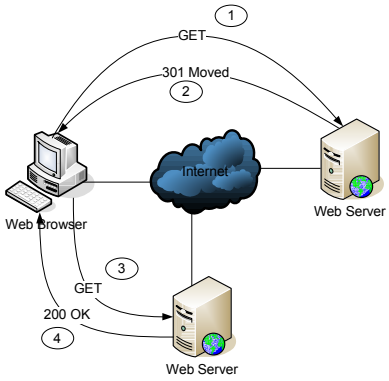
Virtual Host

Solution : utiliser le champ d'entête Host



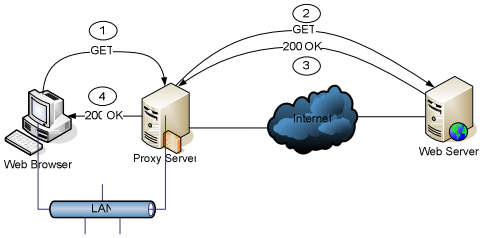
Redirection

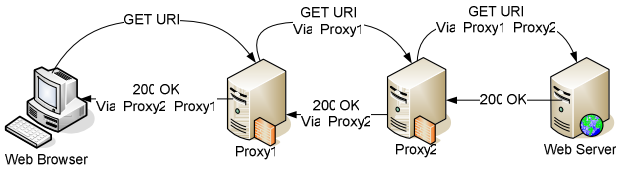
Problème inverse des hôtes virtuels : un site web est présent sur plusieurs serveurs => comment trouver la ressource



Proxy

Cache (Cache Server) + Sécurité (Firewall)

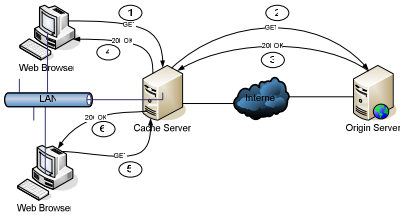




Cache Server

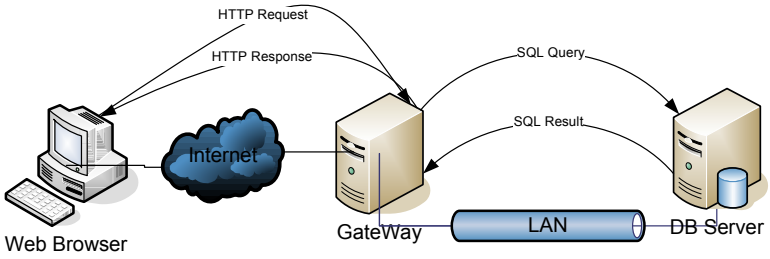
But : optimiser les transferts

Solution : se souvenir des objets précédemment accédés et limiter les accès au serveur d'origine.



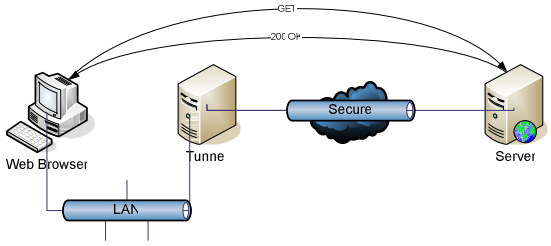
Gateway

Les serveurs Web peuvent utiliser d'autres protocoles d'application, le serveur permettant de passer d'HTTP vers un autre protocole est une passerelle d'application



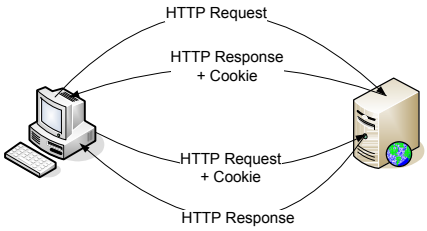
Tunnel

Les tunnels permettent aux navigateurs d'accéder de manière transparente (couches inférieures) à un serveur



Un protocole sans mémoire

HTTP est un protocole sans mémoire, comment maintenir un contexte de session (associer une requête à une autre) => cookies



HTTP

Messages

Méthodes/Status Code

- CONNECT
- DELETE
- GET
- HEAD
- OPTIONS
- POST
- PUT
- TRACE

- 100-199 : Le serveur a bien reçu la requête mais le résultat final n'est pas disponible.
- 200-299 : Succès
- 300-399 : Redirection
- 400-499 : Erreur Client
- 500-599 : Erreur Serveur

Champs d'entête

Les méthodes des requêtes et les codes de réponses peuvent être complétés par des lignes entêtes.

- Requête

```
> GET / HTTP/1.0
> Accept: */*
> Accept-Language: fr
> Proxy-Connection: Keep-Alive
> If-Modified-Since: Tue, 13 Apr 2004 17:45:01 GMT; length=40838
> User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.4322)
> Host: www.free.fr
> Pragma: no-cache
>
```

- Réponse

```
HTTP/1.1 304 Not Modified
Date: Tue, 13 Apr 2004 17:45:56 GMT
Server: Apache/1.3.26 (Unix) Debian GNU/Linux
Connection: close
ETag: "6929-9f86-407c271d"
```

Champs d'entête

Les champs d'entête vont avoir une application :

- Générale
- Pour une requête
- Pour une réponse
- Pour le corps du message

Header Fields

Header	General	Request	Response	Entity
Accept				
Accept-Charset				
Accept-Encoding				
Accept-Language				
Accept-Ranges				
Age				
Allow				
Authentification-Info				
Authorization				
Cache-Control				
Connexion				
Content-Encoding				
Content-Language				
Content-Length				
Content-Location				
Content-MD5				
Content-Range				
Content-Type				

Header Fields

Header	General	Request	Response	Entity
Cookie				
Cookie2				
Date				
ETag				
Expect				
Expires				
From				
Host				
If-Match				
If-Modified-Since				
If-None-Match				
If-Range				
If-Unmodified-Since				
Last-Modified				
Location				
Max-Forwards				
Meter				
Pragma				

Header Fields

Header	General	Request	Response	Entity
Proxy-Authenticate				
Proxy-Authorization				
Range				
Referer				
Retry-After				
Server				
Set-Cookie2				
TE				
Trailer				
Transfer-Encoding				
Upgrade				
User-Agent				
Vary				
Warning				
WWW-Authenticate				

Relecture de notre requête

> GET / HTTP/1.0
> Accept: */*
> Accept-Language: fr
> Proxy-Connection: Keep-Alive
> If-Modified-Since: Tue, 13 Apr 2004 17:45:01 GMT; length=40838
> User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.4322)
> Host: www.free.fr
> Pragma: no-cache
>

- Accept : Type de contenu accepté par le Browser (text/html), ici tous.
- Accept-Language : Langage attendu par le Browser (en-us), ici le français.
- Proxy-Connection : pas dans HTTP/1.1
- If-Modified-Since : Demande au serveur de répondre à la requête que si la ressource a été modifiée depuis la date passée en paramètre.
- User-Agent : Informations sur le client.
- Host : Nom du serveur
- « Pragma : no-cache » : ne pas utiliser les caches mais aller sur le serveur d'origine.

Relecture de la réponse

HTTP/1.1 304 Not Modified
Date: Tue, 13 Apr 2004 17:45:56 GMT
Server: Apache/1.3.26 (Unix) Debian GNU/Linux
Connection: close
ETag: "6929-9f86-407c271d"

- Date : date d'envoi de la réponse
- Server : information sur le serveur.
- Connection : le serveur s'apprête à rompre la connexion.
- ETag : identifiant de ressource

Nouvelle requête

Nouvelle demande sans utilisation du cache local

> GET / HTTP/1.0
> Accept: */*
> Accept-Language: fr
> Pragma: no-cache
> User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.43
22)
> Host: www.free.fr
> Proxy-Connection: Keep-Alive
>
HTTP/1.1 200 OK
Date: Wed, 14 Apr 2004 14:11:40 GMT
Server: Apache/1.3.26 (Unix) Debian GNU/Linux
Last-Modified: Wed, 14 Apr 2004 14:10:01 GMT
ETag: "6a66-9eb4-407d4639"
Accept-Ranges: bytes
Content-Length: 40628
Connection: close
Content-Type: text/html

- Last-Modified : date à laquelle, le serveur d'origine pense que la ressource a été modifiée la dernière fois.
- Accept-Ranges : le serveur permet d'accéder non pas seulement à la ressource entière mais aussi à ses parties.
- Content-Length : Taille du corps du message.

Jouons avec Sioux

Le programme Flicage.java se comporte comme un proxy http et affiche sur la sortie Standard les entêtes HTTP.

Pour l'utiliser, il suffit de configurer le Browser pour qu'il utilise ce client spécifique.

Jouons avec Sioux

- Formulaire GET/POST

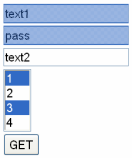
```
<html>
  <head> <title>Test de formulaire</title> </head>

  <body>
    <h1>Test de formulaire</h1>

    <table>
    <form action="200_ok_get_post.cgi" >
      <tr><td><input type="text" name="a"></td>   <tr><td><input type="passwd" name="b"></td>   <tr><td><input type="text" name="c"></td>
        <tr><td><select multiple name="d">
        <option value="A">1  <option value="B">2  <option value="C">3  <option value="D">4
      </select></td>
      <tr><td><input type="submit" value="GET"></td>
    </form>

    <form method="post" action="200_ok_get_post.cgi" >
      <tr><td><input type="text" name="a"></td> <tr><td><input type="password" name="b"></td> <tr><td><input type="text"></td>
        <tr><td><select multiple name="c">
        <option value="A">1 <option value="B">2 <option value="C">3  <option value="D">4
      </select></td>
      <tr><td><input type="submit" value="POST"></td>
    </form>
    </table>
  </body>
</html>
```

Jouons avec Sioux



- Utilisation de get

```
> GET /~remm/CM_HTTP/200_ok_get_post.cgi?a=text1&b=pass&c=text2&d=A&d=C HTTP/1.0
> Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-sho
ckwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, applicati
on/msword, */*
> Referer: http://sioux.src/~remm/CM_HTTP/200_ok_get_post.html
> Accept-Language: fr
> Proxy-Connection: Keep-Alive
> User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.43
22)
> Host: sioux.src
>
HTTP/1.1 200 OK
Date: Wed, 14 Apr 2004 18:48:18 GMT
Server: Apache/2.0.47 (Unix) DAV/2 PHP/4.3.4RC1 mod_jk/1.2.4
Connection: close
Content-Type: text/html; charset=ISO-8859-1
```

Jouons avec Sioux

- Utilisation de Post

```
> POST /~remm/CM_HTTP/200_ok_get_post.cgi HTTP/1.0
> Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-sho
ckwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, applicati
on/msword, */*
> Referer: http://sioux.src/~remm/CM_HTTP/200_ok_get_post.html
> Accept-Language: fr
> Content-Type: application/x-www-form-urlencoded
> Proxy-Connection: Keep-Alive
> User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.43
22)
> Host: sioux.src
> Content-Length: 22
> Pragma: no-cache
>
a=text1&b=pass&c=B&c=D

HTTP/1.1 200 OK
Date: Wed, 14 Apr 2004 18:58:16 GMT
Server: Apache/2.0.47 (Unix) DAV/2 PHP/4.3.4RC1 mod_jk/1.2.4
Connection: close
Content-Type: text/html; charset=ISO-8859-1
```

Jouons avec Sioux

- Cookie Ecriture

> GET /~remm/CM_HTTP/ecriture_cookies.cgi HTTP/1.0
> Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-sho
ckwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, applicati
on/msword, */*
> Referer: http://sioux.src/~remm/CM_HTTP/
> Accept-Language: fr
> Proxy-Connection: Keep-Alive
> User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.43
22)
> Host: sioux.src
>
HTTP/1.1 200 OK
Date: Sun, 18 Apr 2004 21:08:05 GMT
Server: Apache/2.0.47 (Unix) DAV/2 PHP/4.3.4RC1 mod_jk/1.2.4
Set-Cookie: essai=djeff; path=/; expires=Sun, 18-Apr-2004 21:10:05 GMT
Connection: close
Content-Type: text/html; charset=ISO-8859-1

Jouons avec Sioux

- Cookie Lecture

> GET /~remm/CM_HTTP/lecture_cookies.cgi HTTP/1.0
> Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-sho
ckwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, applicati
on/msword, */*
> Referer: http://sioux.src/~remm/CM_HTTP/ecriture_cookies.cgi
> Accept-Language: fr
> Proxy-Connection: Keep-Alive
> User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.43
22)
> Host: sioux.src
> **Cookie**: essai=djeff
>
HTTP/1.1 200 OK
Date: Sun, 18 Apr 2004 21:10:14 GMT
Server: Apache/2.0.47 (Unix) DAV/2 PHP/4.3.4RC1 mod_jk/1.2.4
Connection: close
Content-Type: text/html; charset=ISO-8859-1

HTTP

Authentification

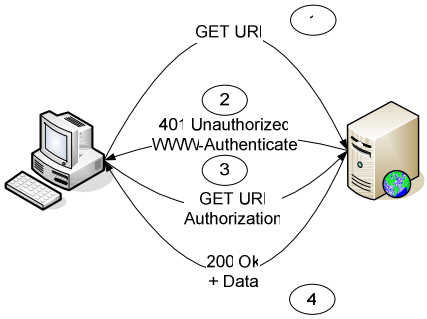
Authentification

- Basée sur user/pass
  - Basic Authentification
  - Original Digest Authentification
  - Improved Authentication
- Dans la pratique utilisation de SHTTP lui-même supplanté par SSL ou TLS

Basic Authentification

1. Demande d'une URI
2. Réponse du serveur avec le champ d'entête WWW-Authenticate: Basic …
3. Réponse avec le champ d'entête Autorisation : user:pass encodé en base 64
4. Envoi de la ressource

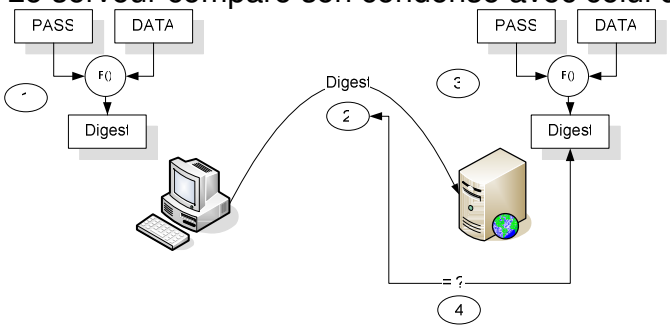
=> User et pass passent en clair



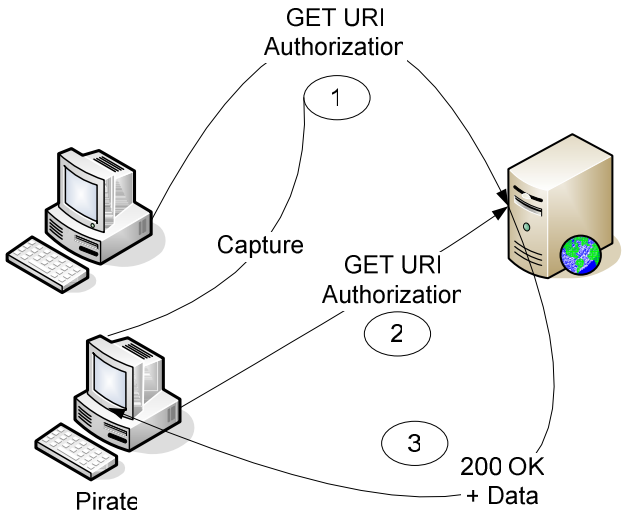
Original Digest Authentication (1.0)

Idem APOP le serveur choisi une donnée (WWW-Authenticate: Digest realm=« … », **nonce**=« data »)

1. Le client calcul le condensé de data et du password
2. Le client envoi sa réponse au serveur
3. Le serveur réalise le même calcul
4. Le serveur compare son condensé avec celui du client



Replay Attack



Improved Authentication (1.1)

- Protection contre les Replay Attack (utilisation d'un incrément nc qui fait parti du condensé)
- Authentification Mutuelle (demande du client au serveur de prouver qu'il connaît le mot de passe.

+ d'autre services (Integrity Protection, Reapeat Client Security)

Conclusion

- Troisième version 0.9, 1.0, 1.1
- Un protocole polyvalent mais complexe
  - Méthodes
  - Format des données
  - Travail des serveurs