Optimisation du transport de contenu
1 - HTTP : proxys et caches

Christophe Deleuze
Grenoble INP – ESISAR

2018–2019

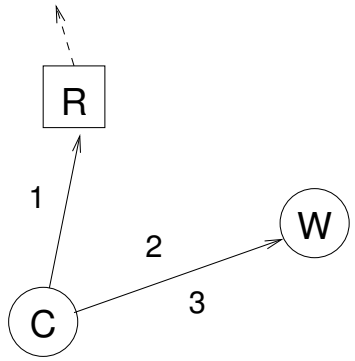
Plan

Plan

Vue de très haut niveau

j'entre une URL `http://www.example.com/coucou.html`

1. le nom est résolu en adresse IP (DNS)
2. une connexion TCP ("tuyau") est ouverte
   1. ⇒ TCP SYN
   2. ⇐ TCP SYN+ACK
   3. ⇒ TCP ACK
3. la requête est envoyée, le serveur répond (en général, gd nb de requêtes et réponses)
4. ... la connexion TCP est fermée



Versions de HTTP

- 0.9
  1. client ouvre connexion et envoie GET coucou.html
  2. serveur écrit le fichier dans la connexion
  3. la connexion est fermée
- 1.0 – rfc 1945
  - format structuré pour les requêtes et les réponses (méta données)

```
⇒ GET / HTTP/1.0
  User-Agent: Wget/1.11.4
  Accept: */*
  Host: www.acrimed.org

⇐ HTTP/1.0 200 OK
  Date: Thu, 08 Apr 2010 13:11:50 GMT
  Content-Length: 8342
  [...]

  [le document]
```

  - différentes *méthodes*

HTTP/1.0 suite

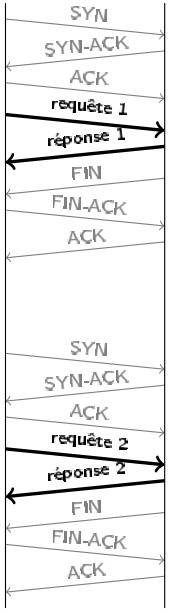
- possibilité connexion persistante (`Keep-Alive:`)

rfc1945 ('96) est *informational*

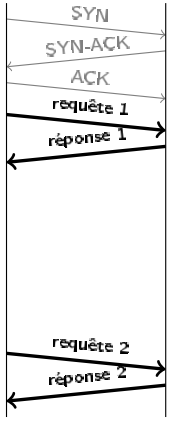
> *HTTP has been in use by the World-Wide Web global information initiative since 1990. This specification reflects common usage of the protocol referred to as "HTTP/1.0".*

Connexions persistantes

Connexions non persistantes



Connexion persistante



HTTP/1.1 – rfc ~~2068~~ ('97) ~~2616~~ ('99) 7230–7235 ('14)

```
⇒ GET / HTTP/1.1
  Host: z80.info
  User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.0.12) Gecko/2
  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
  Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
  Keep-Alive: 300
  Connection: keep-alive

⇐ HTTP/1.1 200 OK
  Date: Fri, 09 Apr 2010 07:50:23 GMT
  Server: Apache/1.3 (Unix) mod_ssl/2.8.28 OpenSSL/0.9.8f AuthPG/1.3 FrontPa
  Last-Modified: Sun, 07 Feb 2010 11:25:56 GMT
  ETag: "35d02c7-490e-4b6ea344"
  Content-Length: 18702
  Connection: close
  Content-Type: text/html

  <HTML><HEAD> [...]
```

Catégories de méthodes (rfc7231)

Safe methods

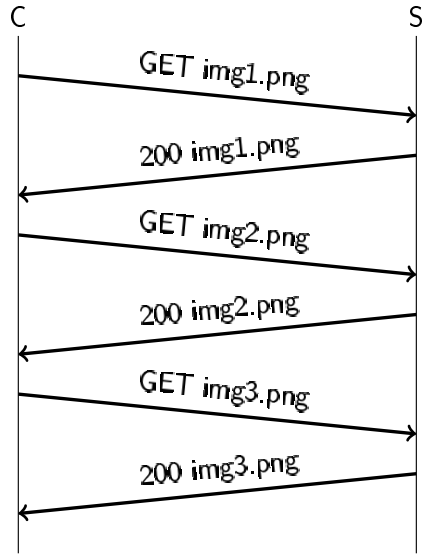
Request methods are considered "safe" if their defined semantics are essentially read-only; i.e., the client does not request, and does not expect, any state change on the origin server as a result of applying a safe method to a target resource.
[...]
Of the request methods defined by this specification, the GET, HEAD, OPTIONS, and TRACE methods are defined to be safe.

Idempotent Methods

A request method is considered "idempotent" if the intended effect on the server of multiple identical requests with that method is the same as the effect for a single such request. Of the request methods defined by this specification, PUT, DELETE, and safe request methods are idempotent

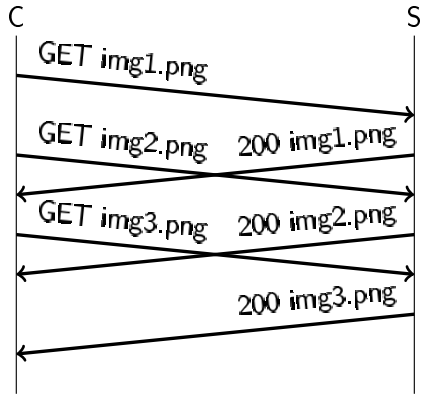
Pipelining

- par défaut, une seule transaction à la fois sur une connexion
  - long si beaucoup d'objets inclus
- une solution : connexions parallèles
  - problèmes d'équité

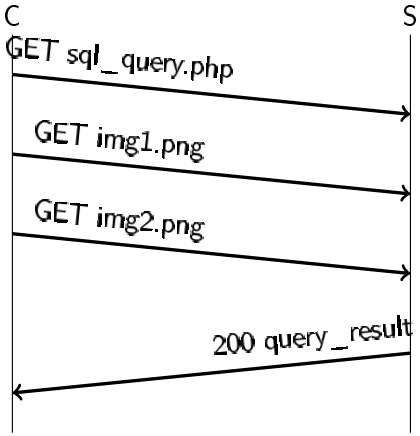


Pipelining

- envoi des requêtes sans attente des réponses
- oui mais :
  - serveur ne gère pas forcément
  - méthodes idempotentes uniquement
  - impossible réordonner les réponses
    - "head of line blocking"



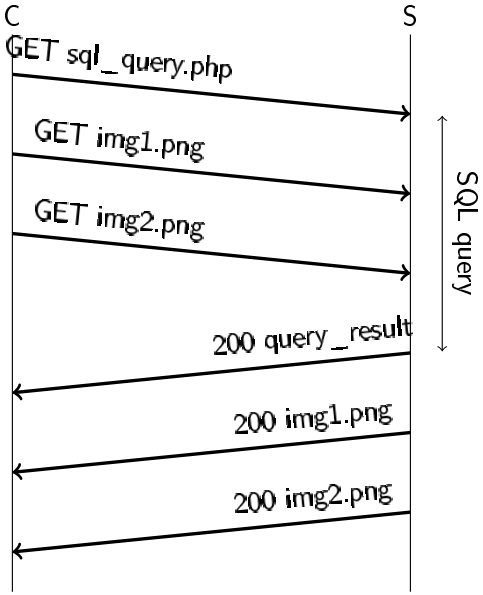
Traitement parallèle



rfc7230-6.3.2+4
A server MAY process a sequence of pipelined requests in parallel if they all have safe methods

Le serveur peut-il traiter les requêtes en parallèle (ou dans le désordre) ?

Head of line blocking



rfc7230-6.3.2+4
A server MAY process a sequence of pipelined requests in parallel if they all have safe methods, but it MUST send the corresponding responses in the same order that the requests were received.

HTTP/2

RFC7540 (mai 2015)

> *This specification is an alternative to, but does not obsolete, the HTTP/1.1 message syntax. HTTP's existing semantics remain unchanged.*

- base de départ spdy (protocole google)
- format binaire avec compression des en-têtes
- multiplexage
- "server push"

améliorations attendues :

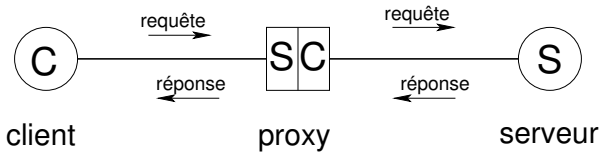
- réduction de la latence
- résolution du problème "head of line blocking"
- amélioration de l'usage de TCP (pas de connexions parallèles nécessaires)

Plan

Proxy – définition générale

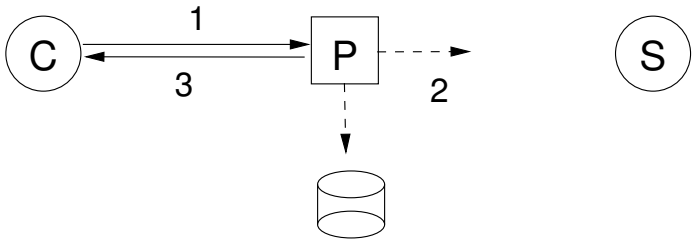
élément intermédiaire dans une communication client-serveur

- passerelle
  - protocoles (gateway)
  - réseau public/privé
- filtrage
- transformation de contenu
- cache



Cache web

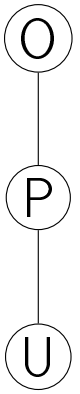
- proxy garde réponses en cache



- gestion du cache
  - *Least Recently Used* (LRU)
  - *Least Frequently Used* (LFU)
  - LRU + taille
  - taille
  - ...

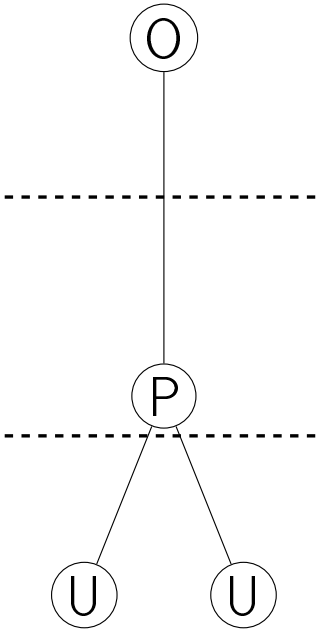
Intermédiaires HTTP

- extrémités
  - le serveur est l'**origine**
  - le client est le **user agent**
- intermédiaires
  - proxy
  - gateway (reverse proxy)
  - tunnel



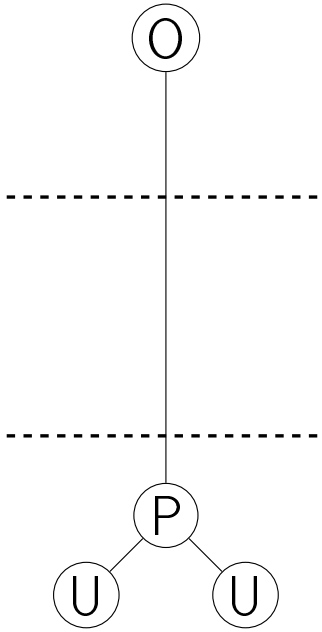
Proxy de FAI

- un *proxy* sert le contenu au client
- géré par le FAI
  - +++ moins de transferts
  - + moins de délai
  - + soulage l'origine
  - – l'origine perd le contrôle (stats, fraîcheur)
- configuration
  - explicite/forcé/transparent



Proxy d'entreprise

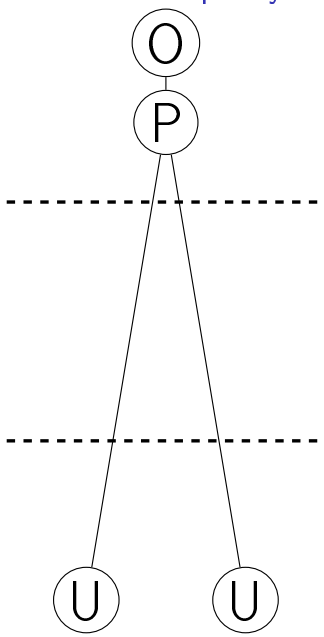
- un *proxy* sert le contenu au client
- géré par l'entreprise
  - + moins de transferts
  - ++ moins de délai
  - +++ applique politique de filtrage (ex interdit facebook/youtube)
  - + soulage l'origine
  - – l'origine perd le contrôle (stats, fraîcheur)



Reverse proxy

- agit comme un serveur origine
- traduit/relaie les requêtes pour le serveur
- géré par le fournisseur de contenu
  - performance (accelérateur web)
  - interface services "legacy" (non HTTP)
  - partitionne/équilibre entre plusieurs serveurs

reverse : par opposition aux deux précédents, appelés *forward Proxy Caching*



Tunnel HTTP

- relaie les messages entre deux connexions
- n'intervient aucunement dans la communication
- un proxy devient un tunnel (pour une communication) par la méthode CONNECT

```
⇒ CONNECT startpage.com:443 HTTP/1.1
  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20140924 conker
  Proxy-Connection: keep-alive
  Connection: keep-alive
  Host: startpage.com

⇐ HTTP/1.0 200 Connection established

⇒ ..............._...&..j..'...v<U,......../n..F.      ← négociation TLS
```


Plan

Support des proxys

- format des requêtes : *cible* de la méthode
  - vers un serveur : *origin-form* = chemin absolu

    ⇒ `GET` */path* `HTTP/1.1`
    `Host: www.example.com`
  - vers un proxy : *absolute-form* = URI absolue

    ⇒ `GET` *http://www.example.com/path* `HTTP/1.1`
    `Host: www.example.com`
- champ `Via`

  ⇐ `Via: 1.1 varnish, 1.0 localhost (squid/3.1.19)`
- méthode `TRACE` et champ `Max-Forward`
- champ `Warning`

Gestion des caches

- quelles réponses cacher ?
- quels éléments des réponses ?
- comment "économiser" en assurant la **cohérence** ? ("transparence sémantique")
  - expiration
  - validation
  - *cache control*

le navigateur typique a un cache local (non partagé)

Quand peut-on cacher ?

les règles sont complexes, en gros ok si :

- la méthode de la requête est connue et spécifiée comme cachable par défaut (GET, HEAD, POST)
- et le code de réponse est connu et... (200, 203, ... , 501)
- et pas de données d'authentification dans la requête

⇒
```
GET /Private/ HTTP/1.1
Authorization: Basic cGhvdG9zOmbsb3BnbG9w
[...]
```

⇐
```
HTTP/1.1 200 OK
Date: Fri, 09 Apr 2010 09:28:26 GMT
Server: Apache/ProXad [Aug  9 2008 02:45:09]
[...]
```

- et aucune indication explicite ne l'interdit (on y reviendra)

ou

- une indication explicite l'autorise

Que peut-on cacher ?

réponse = méta-données + données

hop-by-hop headers : n'ont de sens que sur **une** connexion de transport, ne sont ni cachés ni relayés par les proxys.

```
Keep-Alive
Connection
Proxy-Authenticate
Proxy-Authorization
Upgrade
...
```

plus ceux listés dans `Connection`

Expiration

- date spécifiée par l'origine : `Expires`
- heuristique "adaptive TTL"
  - ex. Cisco Content Engine
    TTL = (CurrentDate – LastModTime) × FreshnessFactor
    (doc décourage FF > 10 %)
  - expiration si age > TTL
    - origine fournit `Date`
    - proxy fournit `Age`

Expiration : calcul de l'âge

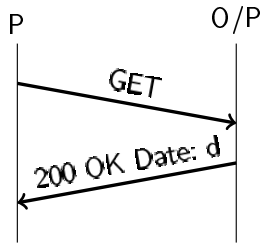
Une réponse servie depuis le cache doit contenir un champ `Age` :

- nb de secondes depuis génération (ou revalidation) par l'origine (temps passé en cache et en transit)

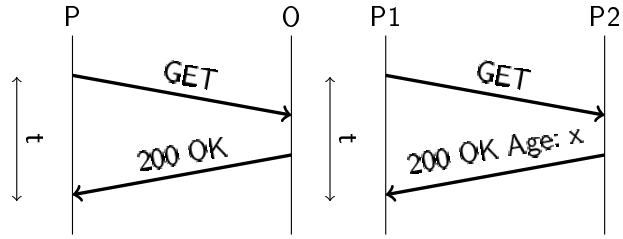
Expiration : calcul de l'âge

proxy doit déterminer l'âge de la réponse qu'il reçoit : deux méthodes

1. utiliser la date de création de la réponse
   âge = max(0, now - d)



2. âge = t
   ou âge = t + x



Validation

- origine associe un "validateur" à la réponse
- req. conditionnelle avec validateur
  - si OK ➠ `304 Not Modified`
- validateurs
  - ⇐ `Last-Modified`
    ⇒ `If-Modified-Since` HTTP/1.0
  - ETag (validateur "opaque")
- validateurs faibles et forts
  - faible : reste valide si chgt "not semantically significant"
    - `GET subrange/204 Partial Content` interdit
  - fort : invalide pour tout chgt
- *browser* reload/shift-reload

Validation : je clique `reload`

```
⇒ GET / HTTP/1.1
  Host: z80.info
  User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.0.12)
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q
  Accept-Language: en-us,en;q=0.5
  Accept-Encoding: gzip,deflate
  Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
  Keep-Alive: 300
  Connection: keep-alive
  If-Modified-Since: Sun, 07 Feb 2010 11:25:56 GMT
  If-None-Match: "35d02c7-490e-4b6ea344"
  Cache-Control: max-age=0

⇐ HTTP/1.1 304 Not Modified
  Date: Fri, 09 Apr 2010 07:51:01 GMT
  Server: Apache/1.3 (Unix) mod_ssl/2.8.28 OpenSSL/0.9.8f AuthPG/1.3
  Connection: close
  ETag: "35d02c7-490e-4b6ea344"
```

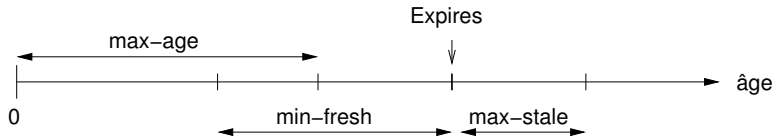
Première requête tout à l'heure...

```
⇒ GET / HTTP/1.1
  Host: z80.info
  User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.0.12) Gec
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.
  Accept-Language: en-us,en;q=0.5
  Accept-Encoding: gzip,deflate
  Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
  Keep-Alive: 300
  Connection: keep-alive

⇐ HTTP/1.1 200 OK
  Date: Fri, 09 Apr 2010 07:50:23 GMT
  Server: Apache/1.3 (Unix) mod_ssl/2.8.28 OpenSSL/0.9.8f AuthPG/1.3 Fro
  Last-Modified: Sun, 07 Feb 2010 11:25:56 GMT
  ETag: "35d02c7-490e-4b6ea344"
  Content-Length: 18702
  [...]
```


Cachabilité

HTTP/1.0 `Pragma: no-cache`
HTTP/1.1 `Cache-Control: ****`



requêtes	**réponses**
`no-cache`	`public`
`no-store`	`private`
`max-age`	`no-cache`
`max-stale`	`no-store`
`min-fresh`	`no-transform`
`no-transform`	`must-revalidate`
`only-if-cached`	`...`
`cache-extension`	*`cache-extension`*

Exemple : je clique `shift-reload`

```
⇒ GET / HTTP/1.1
  Host: z80.info
  User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.0.12) Gecko/20090
  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
  Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
  Keep-Alive: 300
  Connection: keep-alive
  Pragma: no-cache
  Cache-Control: no-cache

⇐ HTTP/1.1 200 OK
  Date: Fri, 09 Apr 2010 07:51:36 GMT
  Server: Apache/1.3 (Unix) mod_ssl/2.8.28 OpenSSL/0.9.8f AuthPG/1.3 FrontPage/5
  Last-Modified: Sun, 07 Feb 2010 11:25:56 GMT
  ETag: "35d02c7-490e-4b6ea344"
  Accept-Ranges: bytes
  Content-Length: 18702
  [...]
```


Cohérence : bilan

support pour :

- cohérence faible
  - "aucune réponse périmée ne survit un temps infini..."
  - OK (adaptive TTL)
- cohérence forte
  - "aucune réponse périmée n'est jamais délivrée"
  - non scalable (revalidation systématique)
- hétérogénéité
  - contraintes par document ou par client
  - OK avec le cache control

manque : invalidation/mise à jour

- délicat à intégrer dans HTTP
- état serveur
- fiabilité/scalabilité
- pb administratifs

Certains fournisseurs n'aiment pas les caches...

```
⇐ HTTP/1.1 200 OK
  Date: Fri, 09 Apr 2010 09:31:34 GMT
  Server: Apache
  Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-
  Pragma: no-cache
  [...]

⇐ HTTP/1.0 200 OK
  Date: Thu, 08 Apr 2010 13:11:50 GMT
  Server: Apache
  Expires: Thu, 19 Nov 1981 08:52:00 GMT
  Vary: Cookie,Accept-Encoding,User-Agent
  Last-Modified: Thu, 08 Apr 2010 11:16:22 GMT
```


Plan

Mais comment configurer ce ... de navigateur !

- explicite
  - manuelle
  - à partir d'un fichier "Proxy Auto Config"
    - fichier js avec fonction `FindProxyForURL(url,host)`
  - forcée
  - automatique
    - eg WPAD
- aucune
  - interception (cache "transparent")

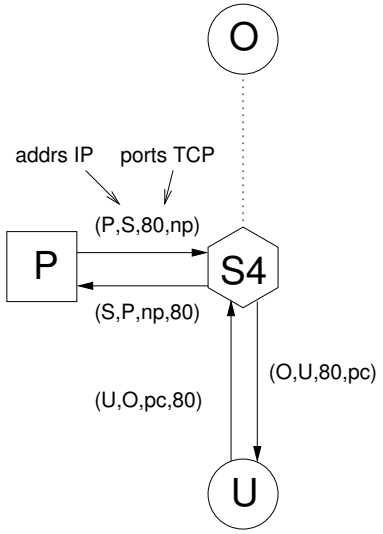
Proxy transparent

- n'est pas sélectionné par le client (aucune config nécessaire)
- *intercepte* les communications HTTP
- usage
  - proxy cache classique
    - obligatoire, sans config
    - applique filtrage ("corporate")
  - portail captif : page de login, ouvre l'accès Internet
    - voir rfc6585

Interception niveau transport

- "switch L4" examine trafic

```
if IP.proto=TCP
   & TCP.flags = SYN
   & TCP.dport = 80
then
  setup_rewrite
  (IP.src, IP.dest,
TCP.sport, 80)
          ↔
   (myIP, proxyIP,
new_port, 80)
```

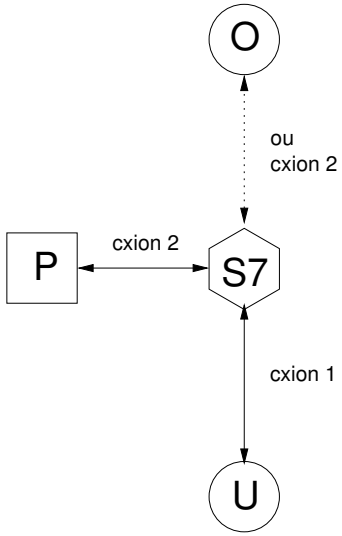


- doit retirer l'entrée quand connexion terminée
- peut rediriger sélectivement sur IP.src ou IP.dst

Interception niveau application

peut se baser sur données application (requête)

- nécessaire ouvrir connexion (prétend être l'origine)
- puis ouvre 2ième connexion avec le proxy (ou l'origine)
- et relaie...



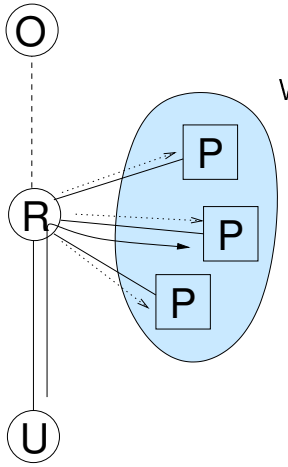
souvent proxy "in-path"

- routeur d'accès
- pare-feu
- NAT
- proxy "transparent"

Pb. avec interception

- pas très catholique
- si pb proxy, accès coupé
- "transparence"
  - authentification/personnalisation
  - recherche DNS inutile
  - ouvre/ferme connexions TCP
  - format requête de type *origin-form*

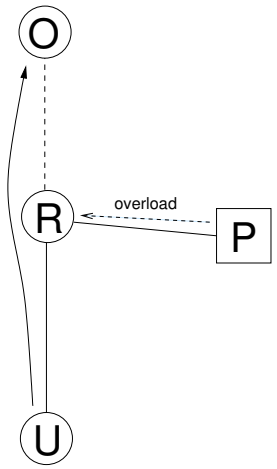
WCCP (Cisco)



Web Cache Communication Protocol

- le routeur intercepte au niveau 4
- gère un cluster
  - hash de l'adr. dest
  - monitore proxys
- plusieurs routeurs peuvent partager le cluster

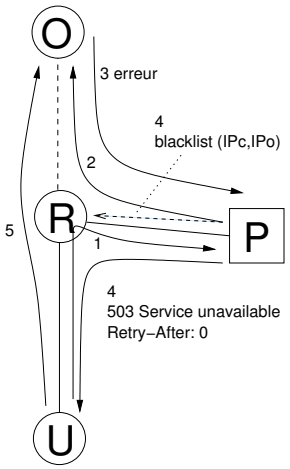
WCCP



overload-bypass

- si le proxy est saturé
- fait suivre les paquets (adr src client)
- réponse en direct

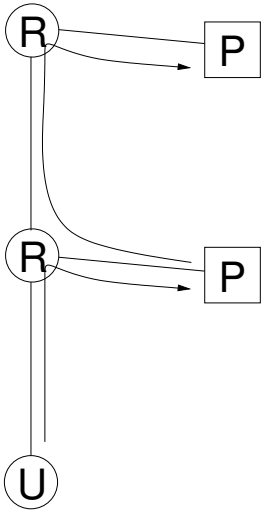
WCCP



dynamic client bypass : si l'intervention (cachée) du proxy crée un problème

1. U émet requête, R intercepte
2. P émet requête
3. O répond erreur (peut être auth sur adr. src ?)
4. P ➠ R : blackliste (IPu, IPo)
   P répond 503 avec `Retry-After: 0`
5. U retente
   R n'intercepte pas

WCCP



gestion d'une hiérarchie
rien à faire de particulier...

On a vu...

... l'usage des proxys "côté client"

- réseau client
- réseau opérateur

Quels usages "côté serveur" ?