Optimisation du transport de contenu
4 - Content Delivery Networks

Christophe Deleuze
Grenoble INP – ESISAR

Décembre 2017

Les fournisseurs de contenu

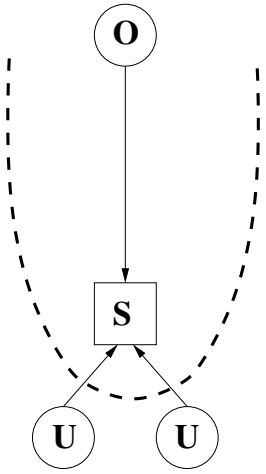
veulent :

- rendre le contenu accessible
  - capacité serveur
  - capacité réseau
  - distribution géographique
- garder le contrôle
  - fraicheur
  - comptage etc
  - web dynamique
  - gestion de site

Content Delivery Network

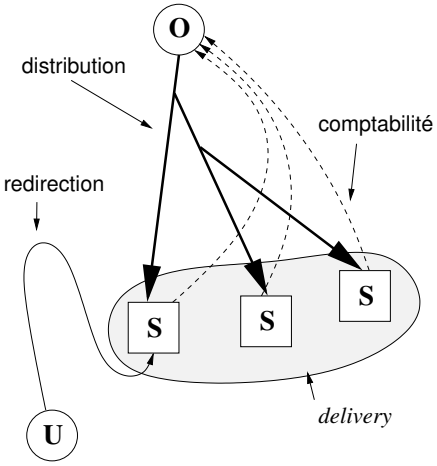
surrogate placé près des clients géré par le fournisseur de contenu

- ++ charge origine
- + débit réseau
- ++ délai client
- ++ stats origine
- ++ fraicheur



Qu'est-ce qu'un CDN ?

- rapprocher le contenu des clients
- quatre systèmes
  - *delivery*
    - serveurs *surrogates*
  - redirection
  - distribution
    - "transport"
  - comptabilité



Qu'est-ce qu'un CDN ?

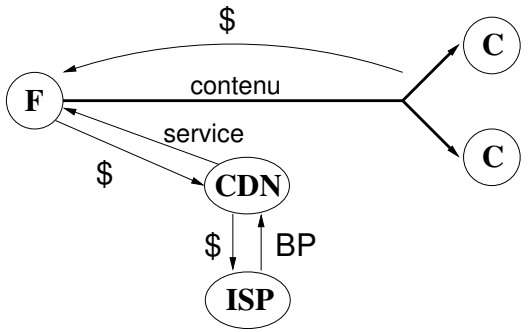
Un content delivery network (CDN) est un fournisseur de services de communications électroniques dont l'une des activités consiste à optimiser l'acheminement de données. Cette optimisation peut concerner aussi bien les performances (latence, débit, etc.) que les coûts d'acheminement. Pour y parvenir, un CDN dispose d'un ensemble de serveurs – dits "serveurs (de) cache" – proches de l'internaute, dans lesquels sont stockés des copies des données à acheminer. Les données en question ne sont donc acheminées qu'une seule fois depuis leur lieu de production jusqu'au serveur cache, ce qui permet d'économiser des coûts de transport / transit ; par ailleurs, la distance d'acheminement dans la partie terminale (depuis le serveur jusqu'à l'internaute) est raccourcie, si bien que les performances sont améliorées. Les clients principaux des CDN sont les FCA. [1]

Rapport au Parlement et au Gouvernement sur la neutralité de l'internet, ARCEP, septembre 2012.

1. Fournisseurs de contenus et d'applications

Qu'est-ce qu'un CDN ?

- fournisseur de contenu
- consommateurs
- opérateur réseau (ISP)
- opérateur CDN



Opérateurs de CDN

- Akamai : 150k serveurs
- Limelight networks
- Level 3 Communications
- Amazon Cloudfront
- Cloudflare
- ...

- services "connexes" de gestion de contenu
  - monitoring
  - reporting
  - DRM
  - identification des utilisateurs
    - spécialiser/segmenter les contenus
  - géolocalisation

Redirection *(request routing)*

- trouver le "meilleur" surrogate (équilibrage de charge + distance au client)
- surrogate = f(consommateur, contenu, état réseau, états surrogates)
- routage de requête
  - routage (construction des tables)
  - relayage *(forwarding)*
- transparent pour le client...
  - différentes méthodes pour le relayage

Redirection DNS hiérarchique

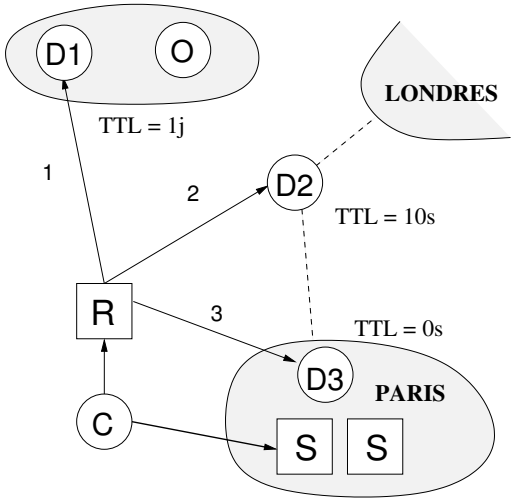
`www.content.com`

fournisseur de contenu

- `content.com NS D1`

opérateur de CDN

- `cdn.com NS D2`
- `paris.cdn.com NS D3`
- `londres.cdn.com NS D4`



Redirection DNS hiérarchique

à D1

⇒ `www.content.com A ?`

⇐ `www.content.com CNAME content.cdn.com`

D2 répond en fct de l'adresse de R

⇒ `content.cdn.com A ?`

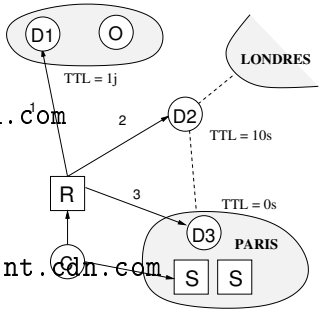
⇐ `content.cdn.com CNAME paris.content.cdn.com`

D3 répond en fct de la charge des surrogates

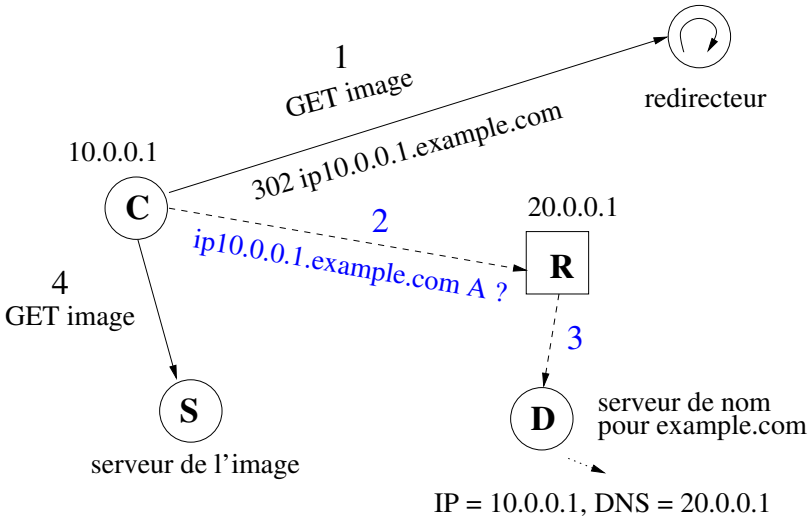
⇒ `paris.content.cdn.com A ?`

⇐ `paris.content.cdn.com CNAME s1.paris.content.cdn.com`

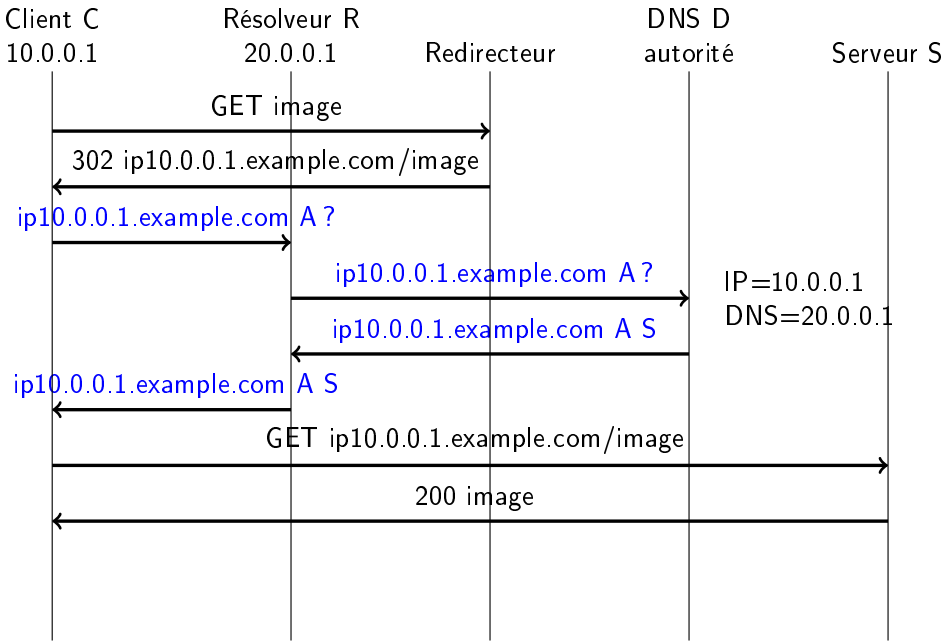
`s1.paris.content.cdn.com A a.b.c.d`



Résolveurs proches des clients ?



Résolveurs proches des clients ?



Résolveurs proches des clients ?

"The Resolvers We Use", The ISP Column, nov. 2014

Résultats (à prendre avec des pincettes)

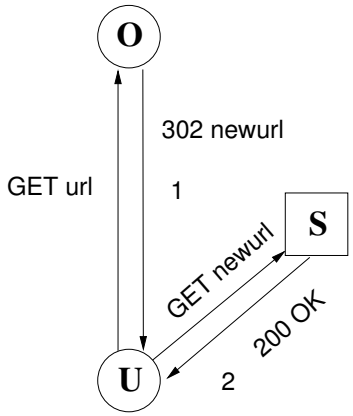
- *Google Public DNS* : 10.5 %, parmi les autres :
- 25 % ont le résolveur dans un autre AS,
- 9 % ont le résolveur dans un autre pays,

Causes possibles

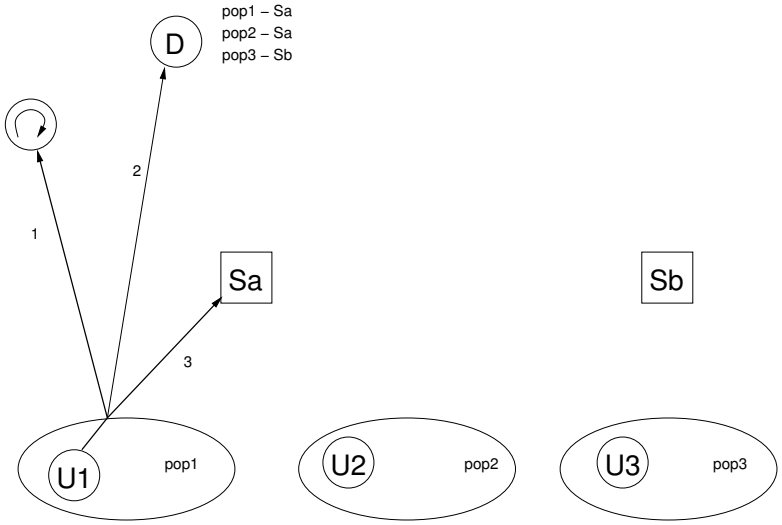
- tunneling pour échapper à censure
- ...

Redirection : *application redirect*

- ⇐ 302 Found
- \+ on voit
  - adresse client
  - URL contenu
- – l'URL du document change !
- – connexion vers l'origine
  - mais une fois suffit...



Réécriture HTTP



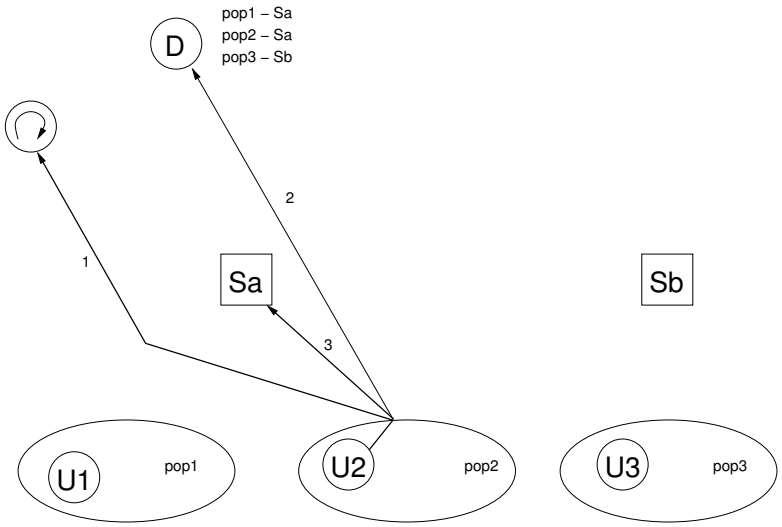
Réécriture HTTP pour U1

```
⇒ GET /content.html
  Host: provider.com
⇐ 302 Found
  Location: http://pop1.cdn.net/provider.com/content.html

⇒ pop1.cdn.net A ?
⇐ pop1.cdn.net CNAME Sa
  Sa A a.b.c.d

⇒ GET /provider.com/content.html
  Host: pop1.cdn.net
⇐ 200 OK
  ...
  <a href="pop1.cdn.net/provider.com/news.png">News!</a>
```

Réécriture HTTP pour U2



Réécriture HTTP pour U2

```
⇒ GET /content.html
  Host: provider.com
⇐ 302 Found
  Location: http://pop2.cdn.net/provider.com/content.html

⇒ pop2.cdn.net A ?
⇐ pop2.cdn.net CNAME Sa
  Sa A a.b.c.d

⇒ GET /provider.com/content.html
  Host: pop2.cdn.net
⇐ 200 OK
  ...
  <a href="pop2.cdn.net/provider.com/news.html">News!</a>
```

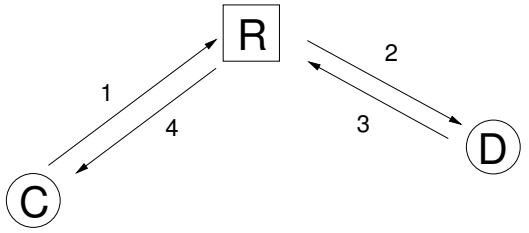

Extension DNS pour localiser le client

une bonne solution

- rfc6891 Extension Mechanisms for DNS (EDNS(0))
  - apr 13, obsoletes rfc2671 aug 99
- Client Subnet in DNS Requests
  - `draft-vandergaast-edns-client-subnet-00` jan 11
  - devenu RFC7871 mai 16 Akamai/Google (informational)
  - permet au résolveur d'indiquer l'adr du client
  - permet au serveur d'indiquer la portée de la réponse

1. résolveur inclut src IP et taille masque dans requête
2. serveur inclut src IP et taille masque dans réponse
3. résolveur cache réponse associée à IP/masque

Extension DNS pour localiser le client



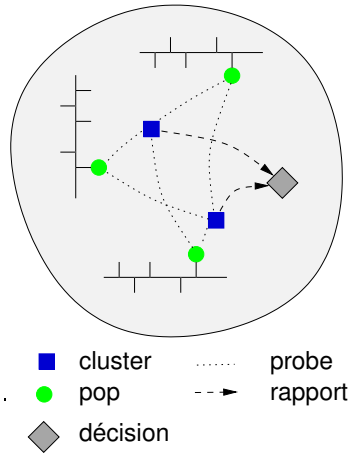
1. `www.example.com A ?`
2. `www.example.com A ? [client-subnet = 192.168.130.1/24]`
3. `www.example.com A 192.168.1.1 [client-subnet = 192.168.0.0/16]`
4. `www.example.com A 192.168.1.1`

Redirection : routage

- requête *client* pour *document*
  - ensemble des Sgtes ayant le document
  - charge (CPU, nb. connexions...)
  - distance au client (hops, délai, débit ...)
- réponse rapide + infos à jour
  - méthode passive (IGP/BGP...)
  - méthode active (sondes)
    - à la demande
    - à l'avance
- Akamai ➠ *secret sauce*
- trouver un *surrogate* raisonnable

Redirection, cas simple : routage intra AS

- approche active/à l'avance
- décision centralisée
- clusters de *surrogates*
- pop (agrégats de clients)
- charge des services dans chaque cluster
- chaque cluster probe tous les pops
- association (pop, service) ➠ cluster
- associations pop ➠ { réseaux } (ex. proto. routage)



Redirection, cas plus compliqué

- construire topologie de l'internet
  - graphe de "réseaux"
    - tout client est dans un de ces réseaux
  - $\neq$ types de liens
    - données historiques et temps réel
  - position des clusters
- ajouter les infos de :
  - charge des clusters
  - classe de trafic
- maintenir le graphe à jour

Source des données

Passive

- infos des protocoles de routage
  - intra-domaine
    - OSPF ➠ topologie
    - RIP ➠ distances en nb de nœuds
  - BGP ➠ AS path length
- traces de trafic
- géo-localisation

Active

latence : echo ICMP (ping)

pertes ...

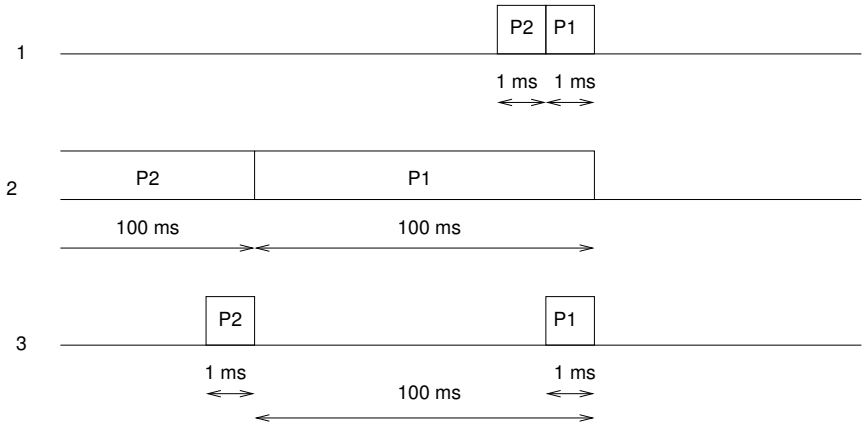
débit
- *packet pair* : débit lien goulot
- *packet train* : débit disponible

Débit d'un chemin : Packet pair I

1. lien à 100 Mb/s
   - j'envoie 2 paquets de 12500 octets, dos-à-dos
   - tps de transmission $\frac{12{,}5\cdot10^3\times8}{100\cdot10^6} = 1$ ms
2. lien goulot à 1 Mb/s
   - tps de transmission $\frac{12{,}5\cdot10^3\times8}{10^6} = 100$ ms
3. retour sur lien à 100 Mb/s
   - les paquets sont espacés...

... mais impact possible d'autres paquets

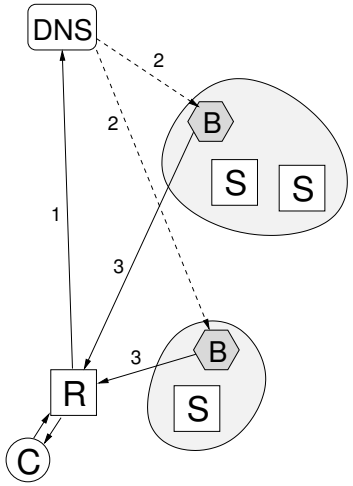
Débit d'un chemin : Packet pair II



Une méthode originale : DNS boomerang

ou Flash DNS ou DNS flooding

- les clusters répondent
  - 1ère arrivée gagne
- proximité réseau
- si serveurs chargés ➠ retarder la réponse
- "routage" très simple !



Distribution : réplication

- Réplication
  - copie
    - *push*
    - *pull*
  - synchronisation (cohérence)

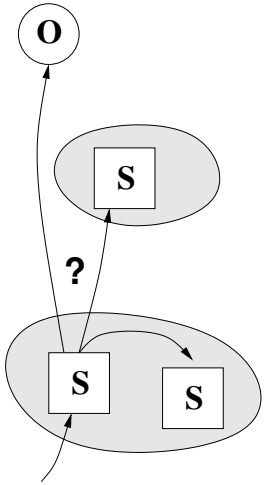
Push

- multicast ?
  - problème de la fiabilité
  - contrôle de flux (hétérogénéité)
  - support du multicast...

Réplication : copie

Pull

- où chercher le contenu ?
  - origine
  - voisins
  - hiérarchie
- forme de routage
  - déterminer les routes possibles
  - choisir la meilleure (distance réseau + charge serveur)



Réplication : ex d'Akamai

hiérarchie de clusters

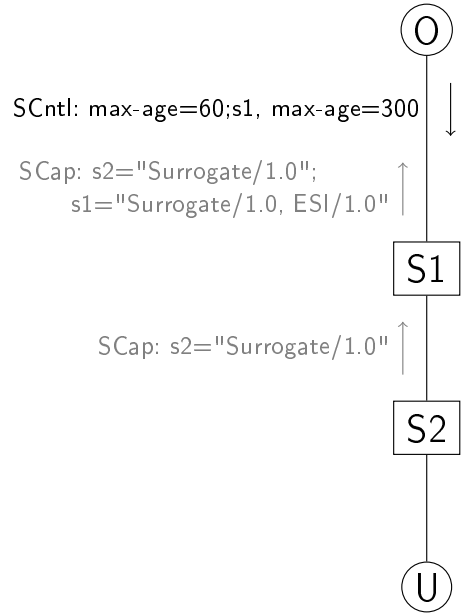
- documents "chauds" dans les clusters feuilles
- documents "froids" dans les parents
  - choisis selon l'état du réseau
- répartition dans chaque cluster
  - optimise espace cache
  - augmente performances
  - pb : dynamicité (charge/crash)
    - *consistent hashing*

ESI - contrôle des surrogates

Edge Side Includes
Akamai/Oracle/... '01

extension au protocole HTTP
(nouveaux en-têtes)

- requête
  `Surrogate-Capabilities:`
  S s'identifie et indique ses *capabilities*
- réponse
  `Surrogate-Control:`
  O indique à S comment traiter le contenu de la réponse



Réplication : synchronisation

maintenir la cohérence des *surrogates* avec l'origine

- *surrogate* = serveur ayant autorité
  - invalidation de contenu
  - mises à jour
  - multicast
- *content signaling*
- tentatives échouées
  - WCIP *Web Cache Invalidation Protocol* '01 (IETF/Cisco)
  - RUP *Resource Update Protocol* '02 (IETF)
- ESI invalidation protocol

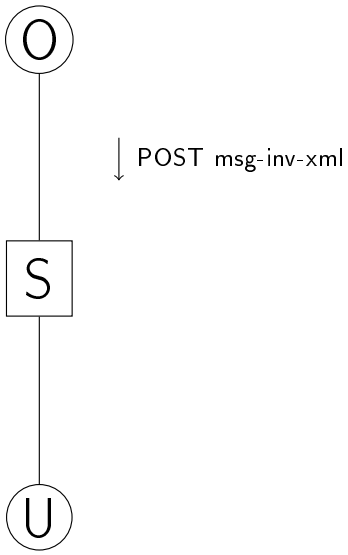
ESI invalidation protocol

- requête HTTP POST avec doc XML (port 4001)

⇐ POST ... *msg-inv-XML*

⇒ 200 OK *msg-result-XML*

- invalidation par
  - URI
  - préfixe
  - regexp URI et en-tête
  - ...




```
O⇒S POST /x-invalidate HTTP/1.0
    Authorization: Basic aW52YWxpZGF0b3I6aW52YWxpZGF0b3I=
    Content-Length: 217

    <?xml version="1.0" ?>
    <!DOCTYPE INVALIDATION SYSTEM "invalidation.dtd">
    <INVALIDATION VERSION="WCS-1.0">
    <OBJECT>
    <BASICSELECTOR URI="/cache.htm" />
    </OBJECT>
    </INVALIDATION>

S⇐O HTTP/1.1 200 OK
    ...
    Content-Length: 284

    <?xml version="1.0"?>
    <!DOCTYPE INVALIDATIONRESULT SYSTEM "invalidation.dtd">
    <INVALIDATIONRESULT VERSION="WCS-1.0">
    <OBJECTRESULT>
    <BASICSELECTOR URI="/cache.htm" />
    <RESULT ID="1" STATUS="SUCCESS " NUMINV="1"/>
    </OBJECTRESULT>
    </INVALIDATIONRESULT>
```

Leases (Baux)

- gestion classique, choix entre :
  - côté serveur (invalidation) ➠ états
  - côté client (revalidation systématique) ➠ msgs
- compromis : *lease* (bail) (obj,S,d)
  - O s'engage à notifier S des modifs de obj pendant d

⇐ `Lease-Control: Grant-Lease | Renew-Lease ...`

- cohérence forte

⇐ `Invalidate-Lease: ...`

⇒ `Invalidate-Ack: ...`

O attend les ACK ou fin du bail pour modifier obj

Distribution de *stream*

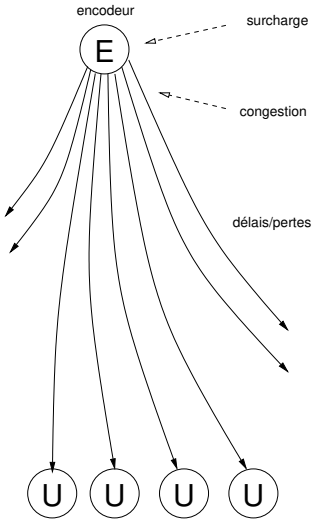
- *stream* : non “élastique”
  - contraintes temporelles internes
  - l'application gère un tampon
  - délai de démarrage acceptable
- stream à la demande
  - juste un fichier...
  - ... oui mais gros
  - ... et occupe longtemps le serveur

➠

  - *prefix caching* puis transfert élastique de la suite

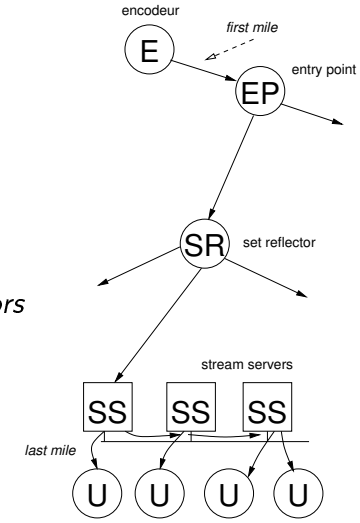
Stream live : les problèmes

- le serveur est surchargé
- son réseau d'accès est saturé
- la traversée de l'internet est longue et dangereuse...



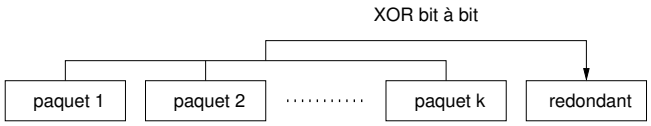
Stream live : architecture

- surrogates sont des serveurs de stream
  - groupés en clusters
  - un élu reçoit le stream et le transmet aux autres
  - last mile
- *entry point* ➠ first mile
- serveurs de transport : *set reflectors*
  - duplication (donc 4 niveaux)
  - contourne routage BGP
  - corrige les pertes



Stream live : qualité du *middle mile*

- pb : perte de paquets
- TCP pas adapté (contrôle de congestion, tempo de retransmission)
- UDP avec ajout correction de perte
  - paquet de redondance
  - tolère un paquet perdu parmi k
  - ➠ redondance temporelle
  - FEC : *forward error correction*



Stream live : qualité du *middle mile*

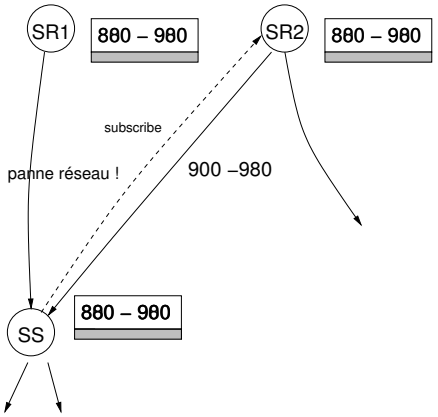
redondance spatiale

- streamer sur plusieurs chemins *(multipath)*
- ➠ "diversité"

retransmission rapide "au mieux"

- émetteur garde les n derniers paquets en tampon
- récepteur envoie NAK si détecte perte
  - reçu n° p+k, toujours pas reçu n° p,
- émetteur retransmet **si tjs dans le tampon**

Middle mile : coupure réseau



- SS élu ne récoit plus
- bascule sur un autre SR
- "prebursting"
  - émettre plus vite au début
  - donc "depuis le passé" (tampon de retransmission du SR)
  - ➠ rattrape les pertes !

Application delivery network

Sites Web dynamiques, applications web...
la plupart des documents ne sont pas cachables

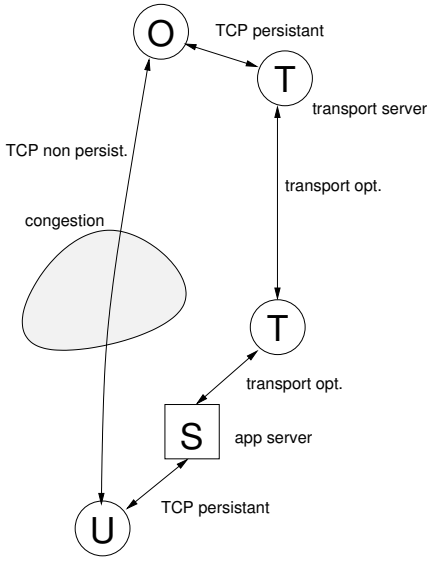
Deux stratégies :

1. optimiser l'acheminement
   - délais
   - quantité données
2. déplacer (une partie de) l'application en bordure

Optimiser l'acheminement

transport **bidirectionnel**
S ne cache pas, mais permet :

- optimisation du chemin (routage à la couche application)
- optimisation du protocole de transport
  - connexions persistantes
  - contrôle de flux/congestion
  - retransmissions rapides
- optimisation de la couche application
  - compression
  - S pré-charge les objets embarqués



Déplacer l'application

ex Akamai EdgeComputing

- plateforme pour exécuter des (composants d') Applications Java2EE

Certaines applications peuvent facilement être déplacées/dupliquées

- aggrégation/transformation de contenus (portails)
- BD statiques (catalogue, recherche sur un site, liste de boutiques...)
- collecte de données (formulaires)

Répartir l'application

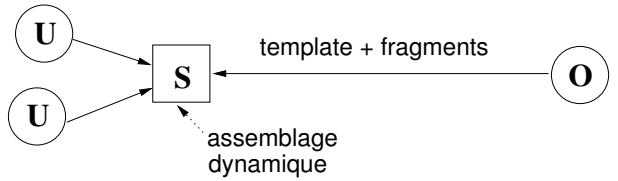
Les applications avec transactions temps réel ne peuvent pas être complètement déplacées.
Archi classique MVC :

- **Model** le cœur de l'appli (BD, règles)
- **View** représentation (sortie)
- **Controller** entrées

- déplacer la partie interface (view et controller)
- réduire les échanges au minimum nécessaire
  - agréger/minimiser les échanges avec le modèle
  - eg l'origine génère une page dynamique qui référence des composants cachables.

Edge Side Includes

- (contrôle des surrogates, protocole d'invalidation)
- assemblage dynamique de fragments en bordure
- fragment
  - paramètres de cohérence
  - partagés par les utilisateurs
- *template*



ESI I

- inclusion (fragments ont leurs propres métadonnées – cachabilité etc)
- variables (basés sur attributs de la requête HTTP, à la CGI)
- traitement conditionnel
- exception, gestion des erreurs (ressources alternatives ou par défaut)

```
<esi:include src="http://example.com/1.html"
             alt="http://bak.example.com/2.html" onerror="continue"/>
<esi:include
   src="http://example.com/search?query=$(QUERY_STRING{query})"/>
```


ESI II

```
<esi:choose>
    <esi:when test="...">
        ...
    </esi:when>
    <esi:when test="...">
        ...
    </esi:when>
    <esi:otherwise>
        ...
    </esi:otherwise>
</esi:choose>
```

ESI III

```
<esi:try>
    <esi:attempt>
        <esi:include src="http://yahoo.com/" />
    </esi:attempt>
    <esi:except>
        Fallback content here
    </esi:except>
</esi:try>
```


Comptabilité

- les logs sont répartis dans tous les *surrogates*
- besoin de générer une vision globale
  - pour le fournisseur de contenu
  - pour l'opérateur de CDN
- scalabilité ?
- structure hiérachique (multicast inverse)

Qui utilise des CDN ? I

```
$ dig www.microsoft.com
;; QUESTION SECTION:
;www.microsoft.com.             IN      A

;; ANSWER SECTION:
www.microsoft.com.      961     IN      CNAME   www.microsoft.com-c-2.edgekey.net.
www.microsoft.com-c-2.edgekey.net. 588 IN CNAME www.microsoft.com-c-2.edgekey.net.global
www.microsoft.com-c-2.edgekey.net.globalredir.akadns.net. 218 IN CNAME e1863.dspb.akamai
e1863.dspb.akamaiedge.net. 1463 IN      A       92.122.180.80

;; QUESTION SECTION:
;www.apple.com.                 IN      A

;; ANSWER SECTION:
www.apple.com.          1373    IN      CNAME   www.apple.com.edgekey.net.
www.apple.com.edgekey.net. 67   IN      CNAME   www.apple.com.edgekey.net.globalredir.ak
www.apple.com.edgekey.net.globalredir.akadns.net. 2792 IN CNAME e6858.dsce9.akamaiedge.n
e6858.dsce9.akamaiedge.net. 879 IN      A       104.93.253.88
```


Qui utilise des CDN ? II

```
;; QUESTION SECTION:
;www.fda.gov.                   IN      A

;; ANSWER SECTION:
www.fda.gov.            1289    IN      CNAME   resolver.fda.gov.akadns.net.
resolver.fda.gov.akadns.net. 1289 IN    CNAME   www.fda.gov.edgekey.net.
www.fda.gov.edgekey.net. 16384  IN      CNAME   e11872.dscb.akamaiedge.net.
e11872.dscb.akamaiedge.net. 1289 IN     A       104.121.26.101

;; QUESTION SECTION:
;blog.lemonde.fr.               IN      A

;; ANSWER SECTION:
blog.lemonde.fr.        148     IN      CNAME   cs205.wac.edgecastcdn.net.
cs205.wac.edgecastcdn.net. 1399 IN      A       93.184.220.239
```

Autre exemple

Plus le cas aujourd'hui...

```
;; QUESTION SECTION:
;www.yahoo.com.                  IN      A

;; ANSWER SECTION:
www.yahoo.com.          535     IN      CNAME   fd-fp3.wg1.b.yahoo.com.
fd-fp3.wg1.b.yahoo.com. 535     IN      CNAME   ds-fp3.wg1.b.yahoo.com.
ds-fp3.wg1.b.yahoo.com. 535     IN      CNAME   ds-eu-fp3-lfb.wa1.b.yahoo.com.
ds-eu-fp3-lfb.wa1.b.yahoo.com. 535 IN   CNAME   ds-eu-fp3.wa1.b.yahoo.com.
ds-eu-fp3.wa1.b.yahoo.com. 708  IN      A       87.248.122.122
ds-eu-fp3.wa1.b.yahoo.com. 708  IN      A       87.248.112.181
```


Google global cache

des serveurs de google installés chez l'ISP, gérés à distance par google

> *Google Global Cache (GGC) enables your company to optimize network infrastructure costs associated with delivering Google and YouTube content to your users by serving this content from inside your network.*
>
> *GGC is implemented as a set of servers deployed in your datacenter, remotely managed by Google. The number of servers deployed will depend on the bandwidth demands of your users and the number of locations at which you chose to install GGC nodes.*
>
> *Google's traffic management system directs users to the node that will provide the best performance for the user.*

Qu'est-ce qu'un CDN ?

Les CDN...

1. partagent dynamiquement l'infrastructure de réplication
2. utilisent les URL classiques
3. fournissent un espace de noms adapté
   - l'URL désigne un document, indépendamment du surrogate qui va le servir

2 et 3 paraissent contradictoires

Un CDN est :

- une infrastructure de réplication
- un changement de sémantique de l'espace de noms URL

Quelques sources

- les RFC mentionnés
- Akamai facts : `http://www.akamai.com/html/about/facts_figures.html`

Nygren (Erik), Sitaraman (Ramesh K.) et Sun (Jennifer). – The akamai network: A platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review*, vol. 44, n 3, 2010.

Kontothanassis (Leonidas), Sitaraman (Ramesh), Wein (Joel), Hong (Duke), Kleinberg (Robert), Mancuso (Brian), Shaw (David) et Stodolsky. (Daniel). – A transport layer for live streaming in a content delivery network. *Proceedings of the IEEE, Special Issue on Evolution of Internet Technologies*, vol. 92, n 9, septembre 2004, pp. 1408–1419.