Copier des sites Web
(Draft)

Conférence B.N.F, 22 Avril 2004

Xavier Roche(HTTrack)

1. De l'intérêt de la copie de sites Web ?

A travers la préservation des sites Web, se profile la problématique de leur copie et de leur stockage. Que ce soit pour des raisons techniques de redondance ou de sécurité, pour des raisons de préservation ou d' « historisation », des raisons légales, la copie de sites Web est un problème rencontré en de nombreuses occasions. C'est d'autant plus vrai qu'elle ne se limite pas au domaine de la préservation, mais concerne aussi bien l'enseignement et la formation – pour du travail sur des copies locales de sites offrant un meilleur confort et évitant la contrainte d'une connectivité à disposition –, la recherche, ou simplement des besoins de particuliers souhaitant conserver leurs sites préférés.

On peut ainsi rapidement classer ces quelques utilisations principales :

- Archivage pour conservation et/ou historisation
- Archivage pour raisons légales (ou établissement de preuves)
- Miroirs de sites pour des raisons de redondance (répartition de la charge et prévention de pannes)
- Copies pour une mise à disposition non connectée (consultation en voyage, lors de conférences, régions ou pays non pourvus de connectivité Internet de qualité…)
- Copies par des particuliers (copie privée) ou des sociétés (CDRom de démonstration)
- Copies pour une analyse technique (liens cassés, plan du site, validation des liens externes) ou linguistique (sémantique, linguistique , « data mining » etc.)
- Agents intelligents (recherche, copie intelligente, « data mining »)
- Utilisations diverses (stress de réseau et/ou de serveur, tests, etc…)

1. Copier un site, qu'est-ce que c'est ?

Cette question n'est pas incongrue ! Pour y répondre, un petit rappel sur la nature des sites Web est nécessaire.

Un site Web peut être vu comme un ensemble de ressources (par exemple, des « pages web » ou des images) accessibles de manière distante, et en général doté d'un système de navigation adéquat (hypertexte). Une attention particulière doit être faite pour ne pas confondre différents concepts liés aux sites Web:

- Le Web n'est pas Internet : il en fait partie, tout comme le courrier électronique, les groupes de discussion, le « chat », etc…
- TCP/IP n'est pas Internet : c'est un protocole de transport d'information, utilisé notamment par http, mais d'autres protocoles sont également utilisés (TCP/UDP, TCP/Ipv6, IPSEC…)
- Les pages HTML ne constituent pas le Web : elles représentent – certes – l'essentiel de la partie hypertexte, mais des sites hypertextes peuvent êtres constitués de fichiers SVG (vectoriels hypertexte), de fichiers flash (Macromédia) etc..
- Le protocole HTPP, associé au « Web », est néanmoins à différencier. Il est la couche protocolaire de transport au dessus de la couche Internet TCP/IP, qui permet de délivrer les informations (ressources) des sites Web. Mais on pourrait imaginer un site qui serait entièrement accessible via FTP (liens `ftp://`) ou en local (liens `file://`) – ce qui nous intéressera pour la consultation de sites Web locaux.

La structure hypertexte, qui permet d'identifier une ressource, et donc d'y faire référence (« lien »), utilise les URL (« uniform ressource locator », soit quelque chose comme « localisateur de ressource unifiée », permettant de localiser une ressource quelque soit le protocole ou l'adresse utilisé)

Par exemple,
`http://www.httrack.com/images/screenshot_01.jpg`

Ici, la partie gauche, `http://www.httrack.com`, représente la localisation du serveur et le protocole utilisé (ici, le protocole est `http`, et la localisation est `www.httrack.com`). La partie droite représente la ressource relative (URI), `/images/screenshot_01.jpg`..

On peut aussi bien faire référence à ces données présentes sur un serveur ftp, et non plus seulement http :
`ftp://ftp.lip6.fr/pub/rfc/rfc/rfc2616.txt`
Ici, nous avons un lien sur un fichier texte, hébergé sur un serveur ftp « anonyme ».

On peut aussi faire référence à des données locales, ou accessibles via un système de fichiers réseau (NFS, SMFS..) :
`file://mnt/disk1/home/smith/report1.ps`

Enfin, des liens peuvent faire référence à des actions, et non plus à des données ; par exemple ce lien particulier qui permet d'écrire à une personne via courrier électronique :
`mailto:smith@example.com`

Nous nous intéresserons plus particulièrement au protocole HTTP, le plus largement utilisé dans le cadre de sites Web.

1.1 « Ressources » HTTP

Le terme de « ressource » pour le protocole HTTP est important : on ne parle pas de « fichier », la notion de ressource allant bien au delà, car une page html hébergée sur un serveur Web peut aussi bien être un fichier (« page statique ») classique stockée sur le serveur, qu'un script produisant du contenu de manière dynamique (« page dynamique ») . La structure même des adresses (URL) Web http peuvent être très différentes des structures classiques de systèmes de fichiers.

Très souvent, on peut facilement établir une pseudo-correspondance entre une ressource hébergée sur un site, et la structure de fichier sous-jacente. Par exemple, pour ce lien particulier :
`http://www.httrack.com/images/screenshot_01.jpg`
on peut imaginer que l'image « `screenshot_01.jpg` » est placée sur le disque dur du serveur Web, quelque part dans un sous-répertoire `/images`. Et c'est effectivement le cas.

Prenons cette autre adresse, qui pointe sur une page en particulier d'un forum de discussion :
http://forum.httrack.com/readmsg/7701/index.html

Si l'on se fie encore une fois à l'analogie avec un système de fichiers classique, on peut imaginer qu'un répertoire « readmsg » est présent quelque part sur le serveur, et qu'un de ses sous répertoires « `7701` » contient une page html nommée `index.html`.
En fait, il n'en est rien. Il existe des milliers de pages dont l'adresse est analogue à celle-ci, mais techniquement, un seul fichier (« script ») en délivre le contenu. Le serveur « sait » que

pour générer la page `http://forum.httrack.com/readmsg/7701/index.html`, un script particulier doit être appelé. Ce dernier « sait » alors comment renvoyer la bonne page, dans le bon fil de discussion, en interrogeant une base de données.

Ainsi, s'il est facile de copier fidèlement l'image du premier exemple, copier fidèlement l'ensemble de ces messages est plus problématique. On ne peut pas accéder à la base de données, et on ne peut pas accéder au « source » du script en question. Et même si cela était possible, il faudrait un programme spécialisé pour permettre à la base et au script de fonctionner convenablement, en les adaptant aux contraintes d'un hébergement local. Dans le cadre de la préservation d'un site, c'est tout bonnement impossible, étant donné la complexité et le nombre infini de systèmes et solutions techniques existant à l'heure actuelle : il faudrait réunir l'ensemble des serveurs Web existant, avec leurs extensions, des bases de données, des langages de programmation, fonctionnant sur des systèmes hétérogènes, des architectures différentes, etc.

Si l'on souhaite copier intégralement la liste de ces messages, il faudra collecter les milliers de pages correspondantes, en interrogeant directement le site Web, et reconstruire une pseudo-structure dans un répertoire local. On n'aura pas alors besoin de programmes spécialisés : les fichiers seront déjà « prêts à l'emploi », disponibles sur disque dur.

Et c'est ici que l'on perçoit deux premières difficultés :

- Des ressources (« pages ») multiples peuvent être délivrées par un seul point central via des systèmes logiques très hétérogènes , et il est impossible de « dupliquer » cette logique tel quel : on ne peut que copier un « exemplaire » délivré à un instant t
- La structure sous-jacente n'a pas nécessairement de réalité effective, et n'est présente que pour des besoins d'identification (de différentiation) des ressources

S'il n'est pas possible de dupliquer la logique délivrant du contenu, mais uniquement de prendre des « photographies », une conséquence évidente apparaît : le côté « dynamique » disparaît, au profit d'un aspect « statique » lié aux fichiers qui ont été construits. Ainsi d'une page Web proposant de donner la température qu'il fait à Paris à tout moment, on ne pourra que prendre un instantané. Si l'on revient une semaine plus tard sur le site, la température aura sûrement changé. Mas pas sa « copie ». Et difficile de copier la logique mise en œuvre : en plus des programmes variés à collecter, il faudrait avoir un thermomètre relié à Paris à tout instant !

Ainsi on peut établir cette conséquence :

> La copie d'un site Web ne peut être qu' un instantané, potentiellement lié au moment de la sa réalisation. Pour un site statique, cet instantané peut se confondre avec sa copie. Mais de la même façon que le cliché d'une scène en mouvement restera figé, la copie d'un site Web dynamique est condamnée à rester « inerte ».

Pour cette raison, la copie d'un site doit être réfléchie : la copie d'un site statique proposant un recueil de nouvelles a un intérêt certain, mais aspirer un site dynamique donnant des températures n'en aura que peu pour le particulier, qui préfèrera effectuer la consultation « en ligne ». Mais pour l'historien ou le statisticien ? Et pour le conservateur ?

2. Copier un site, en pratique ?

Nous avons vu que la copie est plus une tentative de prendre un cliché d'un site à un moment donné, et à préserver ce cliché sur un disque ou une base de données.

En pratique, il faut donc récupérer (télécharger) les pages html correspondantes, qui peuvent avoir été générées dynamiquement, ainsi que les fichiers associés (images et feuilles de style, par exemple), et les sauvegarder. Quoi de plus simple ?

Hélas, il n'existe aucun moyen de « demander la liste » des URLs d'un site Web. Et pour cause : l'existence même d'un lien peut dépendre de la logique mise en œuvre sur ce site ; par exemple de l'existence d'un message dans une base de donnée. L'inventaire des adresses d'un site par ce dernier serait donc une tâche fort peu aisée à réaliser – et finalement fort peu utile pour le commun des mortels.

La seule solution pour collecter la liste des URL à sauvegarder, c'est de tenter de reconstituer cette liste nous mêmes. Comment ? Tout simplement en décortiquant les pages Web – en commençant par une page de démarrage que l'on devra nécessairement connaître – et en collectant les adresses contenues à l'intérieur. Il faudra peut être préciser ce que l'on entend par « adresses contenues à l'intérieur » : beaucoup de sites utilisent des ressources tel que des images ou des feuilles de style hébergées sur d'autres adresses. Certains sites peuvent aussi être répartis sur plusieurs domaines, comme `www.example.com`, `www.example.fr`, `forum.example.com` etc.. Il faut donc procéder au cas par cas, au besoin refaire une copie après avoir identifié les problèmes lors du permier essai.

Prenons un exemple. Dans cette page HTML, qui contient un seul lien :

```
<html>
      <head><title>Page 1</title></head>
      <body>
            Une page on ne peut plus simple
            <a href= "http://www.example.com/page2.html">
                  Page 2..
            </a>
      </body>
</html>
```

On pourra collecter l'adresse « `http://www.example.com/page2.html` ». De même, on procédera de même en collectant les adresses contenues dans les « tags » html « `a` » ou encore «`img` » dans les autres pages.

Mais le lien sera en général donné sous forme simplifiée, « relative », si il fait référence à une ressource disponible sur le même serveur et via le même protocole: `http://www.example.com/page2.html` sera équivalent à `page2.html`, dans la mesure où la page (« Page 1 ») d'origine est disponible sur l'adresse `http://www.example.com/page1.html`

Nous aurons donc également cette forme, plus fréquente :

```
<html>
      <head><title>Page 1</title></head>
      <body>
            Une page on ne peut plus simple
            <a href= "page2.html">
                  Page 2..
```

```
            </a>
      </body>
</html>
```

Deux difficultés supplémentaires apparaissent donc:
- Collecter au sein des pages html les liens contenus dans les « tags » html
- Reconstituer le lien original pour le télécharger sur le serveur correspondant

Ensuite, il faut s'assurer que les liens « fonctionnent » une fois téléchargé. Par exemple, si le site est reconstitué sur un disque local, on pourra modifier la page précédente pour qu'elle ressemble à peu de choses près à :

```
<html>
      <head><title>Page 1</title></head>
      <body>
            Une page on ne peut plus simple
            <a href= "file://C:/My Web Sites/Site1/page2.html">
                  Page 2..
            </a>
      </body>
</html>
```

Mais le lien sera en général construit sous forme « simplifiée » de manière analogue à l'exemple précédent, si bien qu'on pourra écrire :

```
<html>
      <head><title>Page 1</title></head>
      <body>
            Une page on ne peut plus simple
            <a href= "page2.html">
                  Page 2..
            </a>
      </body>
</html>
```

Ainsi, la forme « relative » est identique dans ce cas, qu'elle fasse référence à une ressource disponible sur un site Web, ou sur un disque local. Il faut néanmoins s'assurer que tous les liens sont sous cette forme, et donc les vérifier de manière systématique. De même, si ces liens sont laissés de côté durant la copie, il est préférable de les transformer dans leur version « absolue », de manière à éviter l'apparition de « liens cassés ».

Note : En cas de déplacement du site Web (déménagement, ou site miroir), ou en cas de déplacement du dossier contenant les pages HTML, la forme relative permet un fonctionnement harmonieux quelque soit l'emplacement ou le type de protocole utilisé. C'est cette forme qui est donc en général préférée dans le cadre de copies de sites Web.

Une fois effectuées ces transformations, la page peut être sauvegardée dans une base de données, ou directement sur disque dur, en s'inspirant en général de la pseudo-structure originale : par exemple, `http://www.example.com/page2.html` sera sauvegardé sur disque sous `C:\My Web Sites\Site 1\www.example.com\page2.html`

On collectera de manière analogue – mais sans analyse, non applicable – les fichiers tels que les images, feuilles de style, etc.

2.1 Propagation et préservation de l'information

Ce faisant, nous sommes capables de télécharger les données « tel quel », modifiées au besoin pour des raisons de compatibilité de liens. Mais nous n'avons pas parlé des autres données qui sont envoyées par le serveur : les informations transmises au sein des en têtes http.

Les en têtes http sont une liste d'informations, données en supplément par le serveur http, lorsque vous demandez l'obtention d'une ressource (lien). Ces informations complémentaires (et optionnelles) sont essentiellement la taille de la ressource qui va être délivrée, son « type » (document hypertexte, image, vidéo, fichier archive..), le type d'encodage utilisé si c'est une page html , etc.. De telles informations peuvent être stockées dans une base de données, à part.

Par exemple, pour la ressource « `http://www.httrack.com/html/` », le serveur http associé renverra les méta-données http suivantes :

```
HTTP/1.1 200 OK
Date: Sat, 20 Mar 2004 09:19:01 GMT
Server: Apache/1.3.26 (Unix) Debian GNU/Linux
Last-Modified: Sun, 11 May 2003 13:31:12 GMT
ETag: "10b062e-170e-3ebe50a0"
Accept-Ranges: bytes
Content-Length: 5902
Connection: close
Content-Type: text/html
```

On peut notamment noter :
- la réponse « OK », code[1] 200, indiquant que la demande a été traitée sans problèmes
- le type de ressource, HTML, identifié par son type standard « MIME »[2] (« `text/html` »)
- la taille de la page, de 5920 octets
- des meta-données utiles pour la mise à jour (`Etag` et `Last-Modified`)

Certaines informations peuvent être stockées au sein des fichiers téléchargés :
- la taille du fichier[3] est disponible sur un système de fichiers
- la date de création

Le type de document est plus problématique.

Sur un système de fichier local, le type est indiqué par la manière dont se termine le nom du fichier (« l'extension »), après le dernier « . ». Ainsi, quand vous recevez un document par courriel nommé « `rapport.pdf` », vous allez pouvoir l'ouvrir automatiquement, car le nom du fichier, terminé par « `.pdf` », indique son type, associé au lecteur Adobe Acrobat. Si vous renommez ce document « `rapport.rtf` », il ne pourra plus s'ouvrir correctement : son type a été « modifié » et une autre application essayera de l'ouvrir, sans succès.

Sur un serveur http, le type n'est pas indiqué par l' « extension » : il est indiqué par la meta-donnée « Content-type » envoyée dans les en têtes http. Celle-ci est alors utilisés par les navigateurs pour savoir comment ils doivent traiter la ressource reçue. Bien sûr, sur les sites Web, beaucoup de fichiers HTML ont comme extension « .html », de même que la plu part des images jpeg ont comme extension « .jpg ». Mais ce n'est pas cela qui et utilisé par les navigateurs.

[1] Voir http://www.ietf.org/rfc/rfc2616.txt?number=2616, section 10
[2] La liste exhaustive des types MIME est disponible à cette adresse : http://www.iana.org/assignments/media-types/
[3] Sauf pour les pages de type HTML, modifiées pour les adapter aux restrictions d'une navigation locale

Ce lien, par exemple, pointe vers une page html, et non pas vers une image « jpeg », comme son extension pourrait le faire croire :
`http://www.httrack.com/test_type/images/exemple.jpg`
Si vous sauvegardez cette page sur votre disque, sous le nom « `exemple.jpg` », vous ne pourrez tout simplement pas l'ouvrir !

De même, ce lien pourra être une image de type « jpeg » :
`http://www.example.com/snapshot`
Malheureusement, sauvegardée sous le nom « snapshot » – sans *extension* .jpg – elle sera inutilisable et ne pourra pas être affichée par le navigateur.

2.2 Problèmes de « nommage » des liens

Nous n'avons pas abordé complètement le problème du nommage des fichiers sauvegardés. Dans le cadre d'une base de données, les restrictions sont limitées, et l'on peut raisonnablement garder les adresses originales (URL) pour faire référence aux ressources téléchargées. Lorsque l'on doit stocker ces fichiers sur disque, apparaît le problème épineux de la gestion des noms locaux.

La solution la plus « naturelle », nous l'avons vu, consiste à calquer la structure de fichiers implicite d'un site Web sur la structure locale, en effectuant quelques modifications éventuelles pour indiquer le type de fichier sauvegardé. En clair, cela signifie qu'une page ayant pour adresse `http://www.example.com/private/index.html` pourra être stockée sous la forme `C:\My Web Sites\www.example.com\private\index.html` pour une machine Windows, ou `/home/mirror/www.example.com/private/index.html` pour une machine Linux/Unix. Cette solution est relativement simple et, en permettant de garder la « structure » originale, limite les problèmes de compatibilité avec des scripts javascript ou css qui pourraient implicitement s'y reporter.

Un premier problème concerne la gestion des accents dans les noms de fichiers. En règle générale, les noms de fichiers ne comportent pas d'accents, et la correspondance nom de lien/nom de fichier est simple. Mais lorsque des fichiers en comportent, il faut s'assurer que leur retranscription peut se faire sans problèmes ; et ce, quelque soit le système considéré :

Windows : Système de fichier UCS2 (standard Unicode, 16-bit)
Linux/Unix : Aucune convention ; dépend du système (locale, LC_CTYPE)

Un deuxième problème concerne les problèmes de comptabilité des noms de fichiers, selon le système considéré. Par exemple :

Sur systèmes Windows :

- Les caractères / : * ? " < > | sont interdits
- Des noms réservés, comme aux, nul, lpt1 etc.. sont interdits comme élément de chemin ou de fichier (le nom de fichier « nul.doc » est interdit, par exemple)
- Un nom ne peut commencer par « . »
- Les noms sont insensibles à la casse ; on ne peut créer à la fois « index.html » et « INDEX.HTML »

Sur systèmes Linux /Unix :

- Les fichiers commençant par « . » sont déconseillés (fichiers « cachés »)
- Les caractères tels ~ | * sont déconseillés à cause des comportements spécifiques des « shell » Unix

Dans le cadre d'un stockage sur CDROM :

- 30 caractères maximum par nom de fichier (Joliet ISO9660)
- Les caractères tels * / : ; ? \ sont interdits (ISO9660)

Pour tous ces problèmes, une solution est d'appliquer l'ensemble des restrictions, en remplaçant les caractères interdits (accents interdits, caractères spéciaux..) par le caractère souligné (_), en limitant la taille des chemins et des fichiers, et en ignorant la casse pour la détection des « collisions ».

Mais cette uniformisation des noms peut entraîner des problèmes de collisions, qui peuvent ainsi apparaître dans plusieurs cas :

Adresse Internet	Nom calculé sur disque
`http://www.example.com/index_1.html`	`index_1.html`
`http://www.example.com/INDEX_1.HTML`	`index_1.html`
`http://www.example.com/index:1.html`	`index_1.html`

Dans ce cas, trois pages différentes (avec un contenu différent) risquent d'être sauvegardées sous un même nom une fois copiées. Pour éviter l'écrasement de ces fichiers, et la corruption de la copie du site, on peut par exemple renommer les deux derniers liens en utilisant la convention « nom_2.html », « nom-3.html » etc.. :

Adresse Internet	Nom calculé sur disque
`http://www.example.com/index_1.html`	`index_1.html`
`http://www.example.com/INDEX_1.HTML`	`index_1-2.html`
`http://www.example.com/index:1.html`	`index_1-3.html`

Bien entendu, ces manipulations devront être répercutées au sein des pages html téléchargées, en modifiant parfois certains liens hypertextes. Une conséquence directe, que vous avions déjà entre-aperçue, apparaît ici pleinement :

> Dans la mesure où le type de ressource ne peut pas être stocké avec le fichier, comme peut l'être la date ou la taille, des adaptations sont nécessaires dans le nommage des liens pour assurer un fonctionnement correct lors de la navigation locale.
> Des adaptations sont donc souvent nécessaires pour rendre un site navigable localement ; les liens originaux peuvent alors être perdus. La mise à jour d'un site ne peut donc pas se faire correctement en se basant uniquement sur les données transformées.

3. Et les difficultés apparaissent

A la lumière des explications précédentes, on peut aisément établir la logique d'un aspirateur de sites web à l'aide d'un pseudo-langage :

Enregistrer la première page (hypertexte) dans le tas

Tant qu'il reste des liens dans la pile faire :
    Récupérer le lien sur le tas
    Si la ressource est hypertexte, alors
        Parcourir la page à la recherche de tags « a » et « img », extraire les adresses, les placer sur le tas au besoin, et modifier la page en conséquence
    Fin de si
    Stocker le lien sur disque
Fin de « Tant Que »

En utilisant cette méthodologie, on pourra aisément développer un automate de copie en quelques dizaines de lignes de code tel que le Java ou PERL et récupérer de manière correcte une majorité de sites. Disons 60% des sites. Ou plutôt 60% des liens récupérés correctement.

Mais les autres liens seront cassés, ou manquants. Pourquoi ?

- Il y a beaucoup d'autres tags à analyser (comme <body background= « ..»>), avec parfois des spécificités (fichiers relatifs à un autre répertoire, comme <codebase >)
- Les adresses ne sont pas toujours simples à extraire au sein des tags

Pour un lien tel que :
```
<a href= "page 2.html">
```

- On peut avoir la syntaxe utilisant des guillemets simples :
```
<a href= 'page 2.html'>
```

- On peut rencontrer des caractères « échappés », c'est à dire sous forme codée
```
<a href= "page%202.html">
```

- On peut tout aussi bien ne pas avoir de guillemets
```
<a href= page%202.html>
```

- On peut également rencontrer des « retour chariot » lorsqu'un éditeur de texte a cassé une ligne trop longue, par exemple
```
<a href= "page 2
.html">
```

Le peut aussi être sous forme « absolue », mais également en utilisant des variantes autorisées par la syntaxe des liens hypertexte :

```
<a href= "http:page 2.html">
<a href= "//www.example.com/page 2.html">
```

De même, des caractères « échappés en HTML » peuvent faire leur apparition :

```
<a href= "page 2.html">
```

Mais des erreurs plus génantes peuvent apparaitre:

```
<<a href= "page2
.html">>
```

Ici, le tag lui même est incorrectement écrit.

Ces « cas pathologiques » sont très nombreux, et hélas tolérés par les navigateurs Internet. Cela impose aux aspirateurs de sites de s'y adapter autant que faire ce peux, alourdissant et complexifiant le code de manière très importante.

La « boucle principale » qui détecte les liens et gère ces cas spécifiques représente environ 4,000 lignes de code dans l'aspirateur de sites HTTrack, qui en compte 40,000 environ uniquement pour le « moteur ».

Une fois traités tout ces cas, on arrive péniblement à augmenter le taux de « succès ». Mais beaucoup de liens restent cassés. Continuons notre exploration en profondeur.

3.1 Navigation via des « formulaires »

Certains sites utilisent abondamment les formulaires pour gérer la navigation au sein des pages hypertextes. La grosse différence par rapport à de simples liens, c'est que les paramètres de navigation ne sont plus intégrés dans une adresse (URL) « toute faite », mais dans une liste de champs que le navigateur doit collecter, pour ensuite reconstruire la « véritable » adresse. Disons pour résumer que ce type de technologie est normalement à réserver à ces cas bien définis, comme le remplissage de questionnaires, la composition de courriers, l'envoi d'informations ou de fichiers, etc…

La plupart des aspirateurs de sites web ne gèrent pas par défaut les formulaires, car en règle générale cela apporte beaucoup plus de problèmes que cela n'en résout.
Ainsi certaines « actions » pouvant influer sur l'état d'un site ou de sa base de données sont en règle générale mise en œuvre via des formulaires.

Citons par exemple :
- l'effacement d'un message dans un forum
- l'ajout d'une ligne dans un « livre d'or »
- etc..

De plus les formulaires contiennent potentiellement des entrées que l'internaute doit renseigner lui-même (nom et prénom, par exemple) et que le programme ne peut deviner tout seul.

3.2 Javascript, premier « ennemi » identifié

« Désolé, ce site nécessite un navigateur compatible Javascript pour pouvoir être visité »

Si vous avez utilisé un « vieux » navigateur, comme le vénérable Netscape 2, ou si vous utilisez des outils tels que lynx, ce message vous a peut être été proposé en lieu du site que vous vouliez visiter. Javascript est un langage de programmation adapté aux pages web, executé sur le poste client, et qui permet de dynamiser une page en proposant un nombre important de fonctions. Par exemple, un message d'alerte vous sera envoyé si vous avez rempli de manière incorrecte un formulaire, ou encore un menu « déroulant » vous sera proposé pour faciliter la navigation, ou enfin une animation plus gênante qu'esthétique décorera la page que vous essayez vainement de lire. Le problème avec Javascript, c'est que s'il permet de faire le meilleur comme le pire, il le fait à travers un langage complexe, beaucoup plus difficile à analyser qu'une succession de « tags » html.

Reprenons notre exemple précédent :

```
<a href= "page2.html">
     Page 2..
</a>
```

On peut, via Javascript, créer ce même lien via :

```
<Script language="Javascript">
<--
document.write("<a href= \"page2.html\">Page 2..</a>");
// -->
</Script>
```

De même, on peut charger « dynamiquement » une image en faisant quelque chose qui pourrait ressembler à :

```
<Script language="Javascript">
<--
document.img4.src = "roll0.gif";
// -->
</Script>
```

Un nombre toujours plus important de sites utilise Javascript – pour des raisons plus ou moins légitimes – et une analyse même sommaire du code Javascript devient indispensable.

Cette analyse est lourde, et augmente encore la complexité du code nécessaire à la copie et la collecte des sites. Ajoutons évidemment à cela les « erreurs » de programmation et de syntaxe, souvent tolérées là aussi par les navigateurs. HTTrack comporte une partie importante dédiée à l'analyse javascript, ce qui en fait l'un des rares outils à pouvoir traiter des sites « problématiques ».

Cette analyse ne peut pas être parfaite, car elle supposerait non seulement d'avoir à disposition un évaluateur javascript, mais également un système qui pourrait « intelligemment » comprendre la logique sous-jacente au sein du code, et le modifier pour qu'il s'adapte aux contraintes locales à la manière des « tags » html qui doivent être modifiés pour s'adapter à la forme locale. Evidemment, l'ordinateur intelligent n'est pas prêt d'arriver !

3.3 La liste des formats « ennemis » s'allonge

Autant le dire tout de suite : si l'objectif « 100% de réussite » lors d'une copie de site est irréaliste, s'en rapprocher d'avantage est de plus en plus difficile à mesure que l'on s'en rapproche ! Il est aisé de traiter une grosse moitié des liens, plus difficile d'atteindre les deux tiers, difficile d'atteindre les trois quarts etc.. Plus les efforts sont importants, plus l'analyse se complexifie et plus le nombre de cas augmente, et plus l'on peut augmenter la qualité de la collecte. Mais le gain obtenu est de plus en plus réduit pour un effort de plus en plus important.

Outre les problèmes de tags html retords et de code javasript cauchemardesques, la liste des problèmes est longue.

- Applets java

Tombées un peu en désuétude, les applets java sont de « morceaux de programme » qui s'exécutent sur le poste client, permettant une interaction évoluée, l'affichage de texte, d'images etc. Ce sont des fichiers binaires, comportant du code évolué, donc très difficile à analyser (bien plus que du code « évalué » comme javascript) pour en extraire des liens, et encore plus difficiles à modifier pour les rendre compatibles avec système de fichier local.
HTTrack est l'un des rares aspirateurs à tenter d'extraire les liens. Mais les applets ne sont pas toujours fonctionnelles une fois téléchargées.

- Applets « Flash »

A ranger dans la même catégorie que les applets java, les animations flash sont également des fichiers binaires pouvant comporter des adresses difficiles à extraire. Du code a certes été proposé par le créateur du format Flash pour tenter de répondre à ce problème. Las, ce code est d'une part soumis à une licence restrictive, qui l'empêche d'être incorporé directement dans le cadre de projets libres (ainsi le « plugin » Flash d'analyse pour HTTrack n'est pas disponible sur Debian Linux), mais il est de plus incapable de modifier les liens pour les adapter à une structure locale, et plante allègrement sur un certain nombre d'applets flash disponibles sur Internet ! Enfer en damnation.

- « ActiveX »

Le comble de l'hystérie est atteint avec une version propriétaire des applets précédentes : du code natif (Intel i386) qui ne fonctionne que sur un seul type de système d'exploitation, et qui est là quasiment impossible à gérer. Fort heureusement – si on peut dire –, ces applets « version Microsoft » ayant été massivement utilisées pour diffuser virus et chevaux de Troie, leur existence a fait long feu sur les sites publiques.

3.4 Problèmes liés aux sites eux mêmes

Comme si les difficultés n'étaient pas encore suffisantes, des problèmes non plus liés au format de fichier, mais aux sites délivrant le contenu, apparaissent.

- Sites « à boucles »

Le cas, réel, s'est produit lorsqu'un utilisateurs de HTTrack a tenté de capturer un site en particulier. Le créateur du site avait une section « dynamique », qui n'était pas reconnue comme tel par certains navigateurs, qui s'obstinaient à afficher une vieille version des pages déjà visitées. Pour contourner le problème de manière rapide, la solution technique toute trouvée fut de transformer l'intégralité des liens présents sur les pages dynamiques, pour les rendre « unique à un instant t ».

Pour être plus clair, au lieu d'un simple lien :
`http://www.example.com/page2.html`
le lien devenait :
`http://www.example.com/page2.html?t=19993112235959999`
(ici, le 31/12/1999, à 23h59 :59.999)

Evidemment, en tentant de récupérer la totalité des liens, l'aspiration entra « en boucle », capturant sans cesse les mêmes pages, mais qui changeaient de dénomination à chaque instant. Après une nuit entière de capture, mettant sur les rotules le serveur, le brave ordinateur qui tentait de copier ce site facétieux s'effondra quand son disque dur interne fut saturé.

- Sites « à forte redondance »

Une variante des sites « à boucle » : les sites qui comportent de nombreuses dénominations différentes pour un même lien.

Par exemple, beaucoup de forums de discussion sur le Web permettent d'afficher une page en particulier, identifiée en général par un numéro d'article (ici, 1234)
http://www.example.com/forum/article.php?id=1234

La page suivante n'est pas nécessairement 1235, mais peut aussi bien être 5678 :
http://www.example.com/forum/article.php?id=5678

Le système de navigation peut alors décider de créer des liens « Page suivante » sous la forme :
http://www.example.com/forum/article.php?id=1234&next

Le serveur, lorsqu'il devra délivrer ce lien, saura qu'il doit calculer le successeur de la page 1234, soit la page 5678. Cela permet d'éviter de devoir calculer les successeurs de chaque page lors de la génération d'un lien.

De même, les liens « Page précédente », « 10 pages suivantes », « Début du fil de discussion », etc.. peuvent suivre la même logique.

Une même page (l'article numéro 1234) peut alors avoir d'innombrables dénominations :
http://www.example.com/forum/article.php?id=1234
http://www.example.com/forum/article.php?id=1233&next
http://www.example.com/forum/article.php?id=5678&previous
http://www.example.com/forum/article.php?id=6548&previous10
http://www.example.com/forum/article.php?id=879&next10
etc…

Conséquence : l'augmentation parfois très importante de la bande passante, du temps de copie, et de l'espace nécessaire pour stocker un site donné.

3.5.Autres problèmes techniques divers et variés

Pour les sites :
- Sites protégés par mot de passe (authentification)

Durant le téléchargement :
- En cas d'erreur de transfert ou de connexion interrompue, prévoir une re–tentative de téléchargement
- Gestion des erreurs (liens cassés, par exemple) : stocker la page d'erreur ?
- Gestion problématique des « sessions » (via cookies, par exemple)

- Prévoir de ne pas sauvegarder certains types de fichier (fichiers ZIP, par exemple), ou les fichiers « trop gros »

Type de connexion :
- Utilisation de « proxy » (serveur faisant office d'intermédiaire entre le client et le serveur, notamment pour des problèmes de sécurité) ?
- Copie des liens ftp associés sur certains sites ? (autoriser le ftp sortant)
- Copie de sites en https (sites sécurisés)
- Copie de liens réseau (file://)
- Copie de sites accessibles uniquement sur des domaines IPv6 ? (prévoir connectivité v6)

4. Mise à jour

Lorsque l'on archive un site, ce sont parfois des centaines de méga-octets qui transitent depuis le serveur. L'historisation, ou la mise à jour de ce dernier, ne peut donc pas raisonnablement se faire en re–transférant l'intégralité des données, déjà présentes en grande partie, sauf à gaspiller temps, espace de stockage et bande passante, qui sont tous trois des ressources précieuses.

Fort heureusement, comme nous l'avons entre aperçu précédemment, le protocole http autorise l'utilisation de mécanismes de mise à jour qui permettent – pour peu que le serveur les aient implémentés – de limiter le volume des données lors d'un rafraîchissement.

Deux mécanismes sont principalement utilisés pour permettre la mise à jour de ressources : l'un repose sur l'utilisation d'une date de modification (« `Last-Modified` »), l'autre sur un identifiant plus général (« `Etag` »).

4.1 Mécanisme 1 (« ancienne » méthode, HTTP 1.0[4])

Le principe est simple : le serveur indique, dans les méta-données qu'il transmet au client, la date de dernière modification du document qu'il est en train d'envoyer. Lorsqu'il souhaite « rafraîchir » le document, le client HTPP ajoutera a sa demande la précision « Donne-moi le nouveau document, si il a été modifié depuis telle date »

Par exemple, pour la ressource « `http://www.httrack.com/html/` », le serveur http a renvoyé les méta-données http suivantes :

```
Requête du client           GET /html/ HTTP/1.0
                            Host: www.httrack.com

Réponse du serveur          HTTP/1.1 200 OK
                            Date: Sat, 20 Mar 2004 09:19:01 GMT
                            Server: Apache/1.3.26 (Unix) Debian GNU/Linux
                            Last-Modified: Sun, 11 May 2003 13:31:12 GMT
                            ETag: "10b062e-170e-3ebe50a0"
                            Accept-Ranges: bytes
                            Content-Length: 5902
                            Connection: close
                            Content-Type: text/html
```

.. et notamment la ligne :

[4] http://www.ietf.org/rfc/rfc1945.txt?number=1945

```
Last-Modified: Sun, 11 May 2003 13:31:12 GMT
```

Soit “ce document a été modifié pour la dernière fois le 11 Mai 2003 à 13 heures, 31 minutes et 12 secondes, temps universel »

Lors d’une deuxième visite, le navigateur (ou l’archiveur de sites) précisera donc :

```
Requête du client       GET /html/ HTTP/1.0
                        Host: www.httrack.com
                        If-Modified-Since: Sun, 11 May 2003 13:31:12 GMT
```

Et le serveur pourra alors soit donner une réponse classique si la page a été modifiée depuis, soit répondre par:

```
Réponse du serveur      HTTP/1.1 304 Not Modified
                        Date: Sat, 20 Mar 2004 10:07:29 GMT
                        Server: Apache/1.3.26 (Unix) Debian GNU/Linux
                        Connection: close
                        ETag: "10b062e-170e-3ebe50a0"
```

Soit “cette page n’a pas été mise à jour depuis la date indiquée, vous pouvez utiliser celle que vous avez sous la main”

4.2 Mécanisme 2 (« nouvelle » méthode, HTTP 1.1[5])

Ce mécanisme, beaucoup plus général, est aussi beaucoup plus puissant, car il permet la gestion des mise à jour de ressources pour qui une simple date de modification ne suffit plus : pages dynamiques, dont le serveur ne connaît pas l’historique des changements, pages personnalisées, dépendantes de paramètres régionaux, etc. En principe, ce mécanisme permet de rendre « cacheable » (c’est à dire permettre à un client de ne pas re–transférer l’intégralité des données si elles n’ont pas été modifiées depuis sa dernière visite) toute ressource http, indépendamment de la technologie mise en œuvre derrière[6].

Le principe général est d’indiquer, côté serveur, un identifiant unique (une liste de caractères arbitraires) qui permettra de connaître la « fraîcheur » de la ressource précédemment téléchargée. On pourrait utiliser la date de dernière modification, par exemple, et l’on obtiendrait un système identique au cas 5.1.

5. Précautions associées à l’utilisation de systèmes de copie de sites Web

La consultation d’une page ou d’un site Web est quelque chose de très anodine. Sa copie peut néanmoins poser plusieurs problèmes, indépendamment des problèmes techniques de copie que nous avons pu étudier précédemment.

5.1 Utilisation excessive de la bande passante du site

C’est le principal problème rencontré dans le cadre de l’utilisation personnelle d’un copieur de site Web. Avec l’avènement des connexions domestiques à « haut débit » (qui peuvent atteindre en France 5Mbit/s chez certains fournisseurs d’accès, et beaucoup plus au Japon, où le câblage 100Mbit est une réalité à Tokyo), le taux de transfert potentiel d’un seul client peut

[5] http://www.ietf.org/rfc/rfc2616.txt?number=2616

[6] On pourrait même développer une implémentation de ce mécanisme, qui serait une sur couche abstraite en dessous de la couche HTTP, et indépendante du serveur HTTP

mettre sur les rotules un serveur tout entier[7]. Car s'il est en effet difficile d'abuser de la bande passante lorsque l'on utilise un navigateur classique (à moins de « cliquer » à raison de centaines de fois par seconde !), cela devient fort aisé pour un aspirateur de sites, qui peut facilement analyser des centaines de pages html par seconde, et ouvrir des dizaines de connexions simultanées. Dès lors, le réglage de la consommation de la bande passante est absolument nécessaire pour éviter toute surcharge, qui conduirait le site à prendre des mesures drastiques (interdiction du client, plainte auprès du fournisseur d'accès pour abus, etc..)

Le même problème peut apparaître si vous utiliser la bande passante de *votre propre réseau*, de manière abusive, par exemple dans le cadre d'un réseau d'entreprise, qui peut être facturé au volume. Une saturation de la bande passante peut également perturber le fonctionnement de services critiques, comme l'e–mail, des liens de maintenance sur console, etc.

5.2 Utilisation excessive des ressources de calcul du site

Même en réglant de manière convenable la bande passante utilisée, on peut toujours solliciter de manière trop importante un serveur Web, en lui demandant des pages qui, si elles ne consommeront pas une bande passante trop importante, n'en demeureront pas moins complexes à calculer. C'est le cas par exemple des pages nécessitant des requêtes importantes auprès de bases de données. Une seule page peut demander des dizaines d'opérations complexes sur une base, et la récupération en « rafale » de centaines de pages peut alors ralentir le site entier. Il convient alors de traiter au cas par cas de tels sites, soit en réduisant le nombre de connexions simultanées, soit en diminuant encore la bande passante autorisée, ce qui limitera automatiquement le nombre de requêtes par laps de temps.

Ici aussi, la même question doit être posée pour l'utilisation de ressources qui sont potentiellement partagées (serveur, stockage …) au sein de réseaux d'entreprises ou publics.

5.3 Problèmes légaux liés à la copie elle même

As-t-on le droit de copier un site Web ? C'est une question complexe, car elle fait intervenir plusieurs paramètres :
- Législation en vigueur pour le client (celui qui réalise la copie) et le serveur (pays d'hébergement du site) ?
- Avis du responsable du site (autorisation, interdiction, sans opinion ?)
- Copie privée ? (notion de droit à la copie privée) ou archivage « officiels » (dérogations ?)
- Quelle réutilisation du matériel copié ?

Ainsi, la loi n°95-597 du 1er juillet 1992 (art l 353-3 du CPI)
(..) Est également un délit de contrefaçon toute reproduction, représentation ou diffusion, par quelque moyen que ce soit, d'une oeuvre de l'esprit en violation des droits de l'auteur, tels qu'ils sont définis et réglementés par la loi(..)

De plus, des problèmes supplémentaires apparaissent si l'on souhaite effectuer des traitements automatisés sur les sites collectés. Certains traitements sont explicitement interdits en France, comme la collecte sauvage d'adresse e–mail ou de données personnelles[8].

On peut citer dans la liste des abus :

[7] Par exemple, beaucoup de sites sont hébergés sur des lignes spécialisées (« T1 », « T2 »..) de 1 ou 2 Mbit/s
[8] Loi 78-17 du 6 janvier 1978, code pénal art. 226-16 à 24, et notamment art. 226-18

- La copie partielle ou intégrale d'un site dans le but d'en proposer un « concurrent » (parasitisme, détournement de clientèle via des moteurs de recherche, etc..)
- La copie dans le but de reconstituer une base de donnée mise à disposition (mais dont la copie n'est pas nécessairement autorisée)
- La collecte d'adresses e–mail, dans le but de revente ou de prospection commerciale sauvage (« spamming »)
- La copie massive et répétée dans le but d'occasionner des « dénis de service » (DOS)

etc..

Cependant, la question reste entière pour une utilisation personnelle, dans le but de le visualiser hors connexion. Ici aussi, les avis sont très partagés :

- Oui
- Non
- Il faut demander l'autorisation au « webmaster » du site

Il faut tout d'abord comprendre que le « cache » du navigateur fait déjà, quelque part, le même travail de copie qu'un aspirateur de sites, pour chaque ressource (page, fichier...) visitée :

- Téléchargement des fichiers
- Stockage sur disque
- Réutilisation ultérieure sans nécessiter de connexion

De même, les « proxy » Internet gardent des copies de fichiers téléchargés et les rediffusent à la demande.

Un « aspirateur de sites » se contente finalement d'automatiser la consultation hors connexion du site (on peut le voir comme un robot qui « cliquerait » sur tous les liens du site) en collectant les divers données nécessaires, et en les réorganisant.

Dès lors, il devient difficile de faire la différence entre :

- Un « proxy »
- Un navigateur Internet avec cache
- Un aspirateur de sites web dans le cadre d'une utilisation privée

Je laisse le soin aux juristes de faire la part des choses, entre les exigences de la propriété intellectuelle, et la nécessité de préserver la liberté dans la « sphère privée ».

6. Conclusion

Une fois abandonnée l'idée d'une copie « parfaite », il reste néanmoins des écueils importants selon les sites :

- Complexité du problème, techniquement : nombreux problèmes de copie
- Complexité du problème stratégiquement : que copier ? quelles limites ? intervalle de rafraîchissement ? selon sites ?
- Complexité éventuelle sur les aspects légaux et éthiques (« agressivité » de la copie pour le serveur, données « personnelles » ?)
- Complexité globale de mise en œuvre (ressources nécessaires, en temps de calcul/bande passante/personnes)
- …

La stratégie adoptée par certains projets de préservation est d'utiliser plusieurs systèmes de copie : entièrement automatiques (« recrawl ») comme archive.org, semi-automatiques (aspirateurs de sites), semi-automatiques avec corrections à posteriori (liens cassés corrigés, erreurs diverses…). Il n'y a pas –à priori– de système universel.

Mais cela pourrait changer ?

7. Remerciements

- Julien Masanès (BNF) pour son invitation
- Marie Tétard (ass. Aristote) pour son aide à l'organisation de la conférence
- Ainsi que les nombreux contributeurs et utilisateurs du projet HTTrack !

8. Contact

Le site du projet HTTrack est consultable à cette adresse :
http://www.httrack.com

Remarques, questions et suggestions sont les bienvenues :
Xavier Roche (roche@httrack.com)