INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE
Projet M.I.S.T.R.A.L.

Rapport de D.E.A.

Moteur de recherche d'informations sur le Web

Vincent Morgera

Juin 1997

Encadreur: Monsieur Zhen Liu

1 Introduction

Depuis quelques années, nous avons connu un développement incroyable d'Internet. Créé dans un premier temps, dans un but militaire pour le partage d'informations de façon discrète entre différents points géographiques, il s'est banalisé, et est actuellement, de plus en plus, ouvert à tous. Il offre un certain nombre d'outils, tels que la messagerie ou le transfert de fichiers. Pourtant, pour la majorité des utilisateurs de ce «réseau», l'outil qui caractérise le mieux Internet, et qui a sûrement contribué à son succès, est en fait le WORLD WIDE WEB. Il est vrai que les médias simplifient, la plupart du temps, Internet par cet outil, qui n'est pourtant pas le seul.

C'est ainsi que nous avons suivi le formidable accroissement de cet instrument de diffusion d'informations, communément appelé «Web». Le nombre des serveurs et des sites n'a cessé d'augmenter, multipliant ainsi les données stockées. C'est pour cela, que l'accès à ces informations devient de plus en plus difficile. Ce problème est dû principalement au manque d'organisation initiale de la hiérarchie du Web. Même si une information est présente sur un site, sa consultation n'est pas toujours évidente, voir impossible, si l'adresse de serveur contenant ces données n'est pas connue au départ.

C'est donc, pour répondre à ce besoin de classer les informations du Web, que les premiers moteurs de recherche ont été développés. En fait, ces robots qui parcourent automatiquement les différents serveurs, peuvent avoir d'autres utilités. Mais le principal service reste aujourd'hui l'indexation.

Dans le cadre du stage de D.E.A., une autre utilisation des robots va être faite. En effet, nous allons nous intéresser à la politique de mise à jour des sites Web. Le principe de ce stage est, tout d'abord, de développer un robot qui rapatriera les dates de modification des serveurs, et ensuite, de faire une analyse de ces résultats pour tirer des conclusions sur cette mise à jour des sites étudiés.

Nous allons voir, dans la section suivante, dans quel état sont les recherches sur ce domaine. Une étude du travail effectué sera ensuite donnée dans les chapitres suivants.

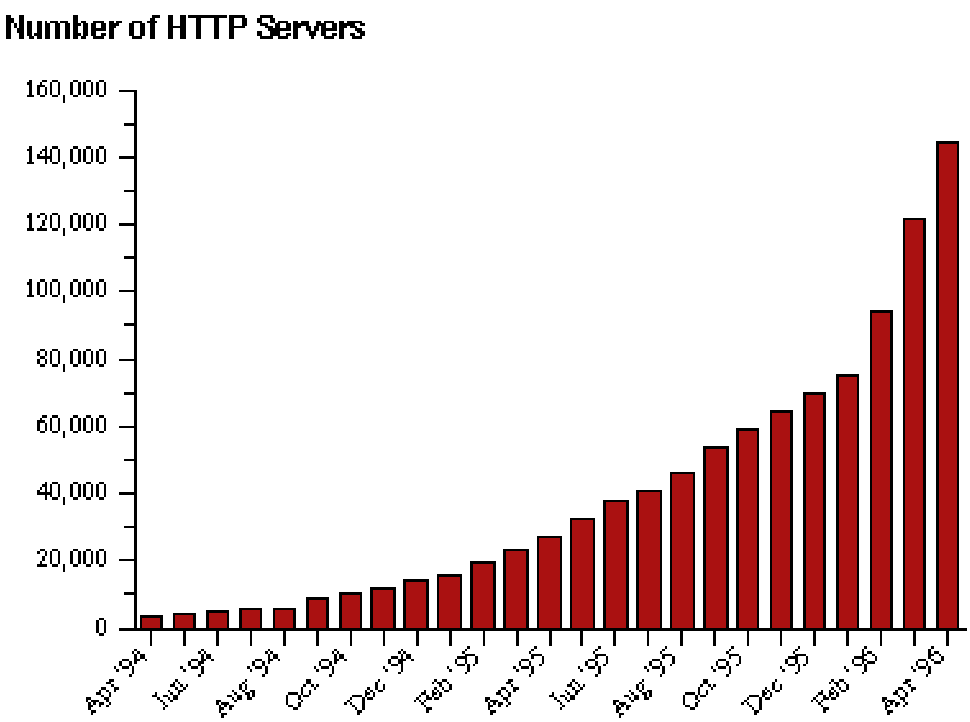
2 Etat de l'art

2.1 Le WORLD WIDE WEB

2.1.1 L'état du WEB

Internet est une interconnection entre réseaux hétérogènes. Un des principals outils connus est le WORLD WIDE WEB, traduit en français par «Toile d'Araignée Mondiale». Créé depuis quelques années, il a connu un gros succès, accompagnant ainsi celui d'Internet. La figure 1 montre cette évolution. Cette comparaison avec une toile d'araignée peut s'expliquer ainsi : Un site est un ensemble de pages HTML [12] référençant d'autres documents. Ces documents peuvent être sur le même site, mais aussi sur des sites distants. On peut considérer ainsi un graphe où chaque site est un noeud et chaque référence entre deux pages une arête. Il est donc possible de se déplacer dans le Web comme dans un graphe en suivant ces liens, et atteindre, n'importe quel autre point, à partir de tout point initial. Cette hypothèse de penser qu'il existe toujours un chemin entre deux points quelconques du graphe n'est qu'une théorie car certains serveurs peuvent ne jamais être référencés par d'autres.

FIGURE 1. L'évolution du Web



Si le Web a connu une telle réussite, c'est sûrement dû à ses possibilités. Une source d'informations peut être transmise à n'importe quelle personne connectée dans le monde entier. Ces informations peuvent s'étendre à de nombreux domaines et donc, beaucoup ont vu un gros potentiel de l'outil. La publicité est l'une des sources de données importantes car c'est un système peu cher vu la clientèle touchée. D'autres domaines comme la distribution, l'éducation ou l'information constituent un ensemble important de sites. Actuellement, le nombre de serveurs ayant énormément augmenté, les sujets se sont d'autant plus diversifiés. Il est donc possible d'accéder aujourd'hui à n'importe quel thème voulu.

Mais, cet accroissement, non prévu, a vite posé certains problèmes. La plus grosse difficulté a été justement cet accès aux informations. Le Web a grossi si vite, sans aucune organisation, ni contrôle, que trouver une information précise s'avère une tâche bien difficile. Cette croissance dynamique a provoqué une *anarchie* dans le Web. Aussi, a t-il été important de résoudre ce problème afin de rendre l'utilisation du Web aussi conviviale que pratique. L'idée d'indexer le Web [1] est apparue comme la meilleure alternative pour accéder aux informations contenues. Cette solution est, dans un premier temps, de parcourir les différents sites et de ranger ensuite les informations collectées dans des banques de données. Automatiser ce travail s'est donc avéré indispensable. Et, le développement des premiers moteurs [5] de recherche s'est fait afin de répondre à cette demande.

2.1.2 Le travail des moteurs de recherche

Historique

WebCrawler [11] a été un des premiers moteurs de recherche lancé sur le marché. Développé dans un projet de recherche de l'Université de Washington au début de l'année 1994, il a été une véritable réussite commerciale. Au début de son exploitation, la base de données de WebCrawler contenait des informations sur 6000 serveurs WEB. Son succès en a fait un outil très vite utilisé et dès la fin 94, le service recevait plus de 15000 requêtes par jour. Depuis, de nombreux autres moteurs ont été mis en circulation : Lycos, InfoSeek, Open Text, Yahoo, etc ... Le nombre de ces moteurs est aujourd'hui très grand, la plupart n'étant pas officiels, mais seuls certains sont très souvent utilisés.

Le vocabulaire utilisé

Pour la suite de ce rapport, nous utiliserons aussi bien le terme de moteur de recherche que celui de robot. Nous retrouvons également dans les ouvrages anglais les mots Spider, Worm ou Web Crawler (ne pas confondre avec WebCrawler (vu auparavant) qui est un robot spécifique).

En fait, on parle de moteur de recherche, ou de robot, pour un programme qui fait des recherches sur un ensemble de données. Dans le cas du World Wide Web, C'est un automate qui va utiliser le contenu des pages HTML comme données pour faire ses recherches. A partir de ces documents, le robot va stocker des éléments qui pourront être réutilisés au moment de l'interrogation. L'utilisation d'un moteur de recherche ensuite, n'est en fait qu'un accès, interfacé par une page Web, à une base de données préalablement remplie. Le robot, lui, est lancé de façon invisible aux utilisateurs à des dates périodiques pour maintenir ces tables à jour.

Le travail

Le moteur de recherche travaille de façon simple. On peut le comparer à un browser HTML tel que **Netscape**, mais contrairement à lui, la navigation se fait de façon automatique. C'est à dire, à la différence des navigateurs habituels, sur lesquels le passage de document en document nécessite la participation d'un utilisateur qui doit choisir le lien à suivre, le robot va suivre toutes les pages référencées par une page initiale et cela de façon récursive.

Le moteur de recherche démarre toujours sa recherche sur une liste d'adresses. Cette liste est nécessaire pour pouvoir explorer tout le Web et pour éviter l'oubli de certains sites non référencés par d'autres serveurs. Il est donc possible de demander à rajouter l'adresse de son propre site si l'on désire rendre disponible ses données à partir d'un moteur de recherche. Ensuite, le travail se déroule ainsi. Pour chaque page initiale de la liste, le robot va faire une requête HTTP (protocole décrit dans une section suivante) et récupérer ainsi un ensemble de données, dont le contenu du document HTML. Il va analyser tout ce document afin d'en extraire les différents liens qu'il peut contenir. La plupart des pages sont référencées en HTML de la façon suivante:
<A HREF="page.html">texte</A>
Pour chaque nouvelle adresse obtenue, le robot vérifie qu'il ne l'a pas déjà visitée plus tôt, et si ce n'est pas le cas, alors il procède de la même façon sur cette page et cela récursivement jusqu'à ce qu'il ne trouve plus aucunes nouvelles adresses à visiter. On peut considérer le travail d'un robot comme le parcours d'un arbre. Les boucles sont évitées en vérifiant à chaque fois que l'on ne visite pas deux fois la même adresse. Cette visite se fait en largeur puis en profondeur. Nous verrons un exemple de parcours d'arbre dans la section 3.1.2 «Le parcours du robot».

Quelques soient les robots, le parcours se fait à peu près de façon similaire. Après cela, chaque robot a un travail spécifique. Dans le cas de l'indexation du WEB (qui était le but premier des moteurs de recherche), pour chaque adresse visitée, le robot va stocker des données liées au contenu du document : titre, mot clef, paragraphes, etc ... Les informations seront ainsi rangées dans une banque de données. Une fois ce stockage fait, la recherche des pages correspondant à un sujet ne sera qu'une requête sur la banque.

L'indexation

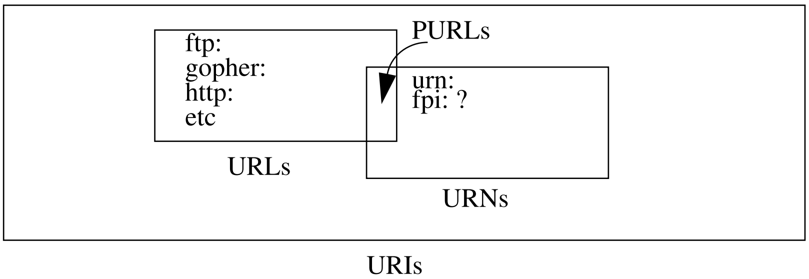
Les techniques d'indexation sont diverses et variées. Au début, les titres des documents servaient comme mots clef de recherche, mais cette solution a vite montré ses limites. Le titre d'un document reflète très rarement le contenu du document. Donc la solution adoptée est de stocker non seulement ce titre mais aussi tous les mots des premiers paragraphes. Dans les versions récentes de HTML, un nouveau tag a été créé pour permettre à l'auteur de la page de spécifier les mots sur lesquels il désire faire apparaître les recherches. Mais ce tag n'est pas encore assez propagé pour être vraiment utilisé par les robots et la technique d'indexer le Web par rapport au contenu et au titre du document reste toujours la solution suivie. L'idée du tag reste tout de même une solution d'avenir afin de vraiment déterminer plus précisément les mots clef dès la création de la page.

2.2 Les services du WEB

2.2.1 Les adresses URI

Les URI (Uniform Resource Identifiers) sont les moyens pour représenter toutes les adresses des ressources (pages, services, documents, ...) sur le Web.
Les URL (Uniform Resource Locator) sont un ensemble d'URI qui possèdent les instructions explicites pour accéder à toutes les ressources sur Internet. Dans leurs formes absolues, elles peuvent contenir un certain nombre d'informations comme la localisation ou une partie du chemin de la ressource. Dans le cas du protocole HTTP, c'est ce type d'ensemble qui sera utilisé.
Les URN (Uniform Resource Name) sont un autre ensemble d'URI que nous ne détaillerons pas.
Le graphe suivant donne une représentation de ces ensembles.

FIGURE 2. UNIFORM RESOURCE IDENTIFIERS



2.2.2 Le protocole HTTP

Le protocole HTTP (HyperText Transfert Protocol) [3] est un protocole de transfert utilisé par le système hyper-média WWW afin de récupérer des informations à partir de serveurs distribués. Ce protocole utilise lui même le protocole TCP comme moyen de transport. C'est donc une couche placée au dessus de cette couche transport. Le problème est, que certaines caractéristiques particulières à HTTP agissent mal avec TCP ce qui provoque des problèmes de performance et de débit. En fait, le problème du protocole HTTP est d'utiliser une connection pour chaque requête.

Par contre, ce protocole est très simple. Le client ouvre une connection et envoie une requête. Le serveur exécute la requête en question, retourne le résultat puis ferme la connection établie.

Nous allons déterminer plus précisément comment un client récupère des données HTML sur un serveur HTTP à partir d'une adresse.

Le client établit une connection avec le serveur (dont il connaît l'adresse) et lui envoie une requête. Cette connection se fait sur le port 80 si le port n'est pas précisé explicitement ou sur le port donné. Une adresse HTTP se décompose ainsi :

HTTP://hostname[:port]/path[?searchwords]
- hostname : nom du serveur (numérique ou alphabétique)
- port : numéro de port (facultatif)
- path : chemin où se trouve le fichier (à partir du chemin initial des documents HTML)
- searchwords : paramètres pouvant être passés à la page

Les requêtes envoyées sont également simples. C'est une chaîne de caractères ASCII terminée par le caractère CrLf (line feed). Il faut ensuite juste passer le type de méthode utilisée (get ou post) (comment passer les paramètres), l'adresse du fichier voulu (en omettant http:) et d'autres informations sur le client.

Syntaxe d'une requête

```
Request          =   SimpleRequest | FullRequest
SimpleRequest    =   GET <uri> CrLf
FullRequest      =   Method URI ProtocolVersion CrLf
                     [*<HTRQ Header>]
                     [<CrLf> <data>]
<Method>         =   GET | POST | HEAD | ... /* mais les deux premières sont
                                    pratiquement les seules utilisées */
ProtocolVersion  =   HTTP/1.0
<HTRQ Header>    =   <Fieldname> : <Value> <CrLf>
<data>           =   MIME-conforming-message
```

Exemple de requête

```
GET  /index.html HTTP/1.0
Accept: text/plain
Accept: text/html
Accept: */*
User-Agent:  Yet Another User Agent
```

La réponse est également une chaîne de caractères ASCII. Cette chaîne contient la version HTTP utilisée, des données descriptives du document (taille, date de modification...), et les données HTML (HyperText Mark-up Language) du fichier demandé.

Syntaxe d'une réponse

```
HTTP/1.0 <result-code> [<message>]  <crlf>
{<Header>: <Value> <crlf>}*
<crlf>
```

Exemple de réponse

```
HTTP/1.0 200 OK
Server: MDMA/0.1
MIME-version: 1.0
Content-type: text/html
Last-Modified: Thu Jul  7 00:25:33 1994
Content-Length: 2003
<html><body>
<title>MDMA - Multithreaded Daemon for Multimedia Access</title>
<hr>
</body></html>
```

Si un document demandé n'est pas trouvé, (ex: document inexistant ou protégé en lecture), un fichier standard est envoyé au client qui indique le type d'erreur.

La connection TCP-IP est cassé par le serveur une fois que celui-ci a fini de transférer toutes les données du document au client. Le client a également le droit de stopper un transfert en cours en cassant lui-même cette connection. Dans ce cas, le serveur ne doit pas prendre cette fin de connection comme une erreur de transfert et tenter de renvoyer les données.

Revenons sur le problème des performances : en faisant des test sur le protocole HTTP, on s'apercevrait qu'il passe la plupart du temps à attendre (mise au point de la connection) au lieu de transférer des données.

Pour éviter ce problème de performance, l'idée est de ne créer plus qu'une seule connection (phase longue dans la communication client/serveur) et de l'utiliser pour réaliser plusieurs requêtes. Surtout que l'on s'aperçoit que les pages référencées par une page initiale sont la plupart du temps sur le même serveur. Ce protocole HTTP-NG est désigné pour remplacer le protocole HTTP 1.0. Il n'est plus une couche qui se place au dessus de TCP mais qui remplace la couche transport.

2.3 Les recherches faites jusqu'ici

2.3.1 Le robot mathématique Pérégrinator

Développé à partir de juin 1994, Pérégrinator [7] s'est inspiré des recherches de WebCrawler (voir la section suivante). Il n'est pas en accès libre car l'auteur précise que son robot était un robot «impoli». La politesse, [9] et [10], au niveau des robots est l'occupation du réseau qu'ils peuvent prendre sur un serveur interrogé. Pour éviter de surcharger le serveur interrogé, il faut avoir une politique de visite et limiter le nombre de requêtes envoyées dans un délai donné. Bien que la réponse à une requête soit relativement rapide (comparée au chargement d'un document avec un navigateur traditionnel qui doit charger en autre les images), il faut brider le parcours afin de ne pas encombrer le serveur distant qui accepte d'autres utilisateurs en même temps.

Pour l'indexation, ce robot utilise certains tags HTML. Il stocke dans sa base comme mot clef tous les mots compris entre *<TITLE>...</TITLE>* ou *<H1>...</H1>*. Le tag <H1> indique une grosse police d'affichage ce qui implique que ce texte doit être le plus important du document.

2.3.2 Le moteur de recherche Webcrawler

Beaucoup de recherches ont été faites dans le domaine des moteurs de recherche. Mais le développement de WebCrawler [11] reste une référence. Martjin Koster a montré quelles étaient les différentes utilités des robots. Ils peuvent être utilisés, tout d'abord, pour pouvoir faire des études statistiques (compter le nombre de serveurs existant par exemple). C'est dans cette optique que le moteur de recherche lié au stage sera développé. les robots permettent également la maintenance des structures HTML. Ils peuvent être également développés pour construire des sites miroir (copie intégrale d'un site distant en local afin de minimiser ensuite les temps d'accès). Bien que cette technique paraisse très intéressante pour diminuer les temps d'accès réseau, aucun outil n'a vraiment été développé à ce jour. La recherche d'informations reste en fait la principale utilisation des robots sur le WEB.

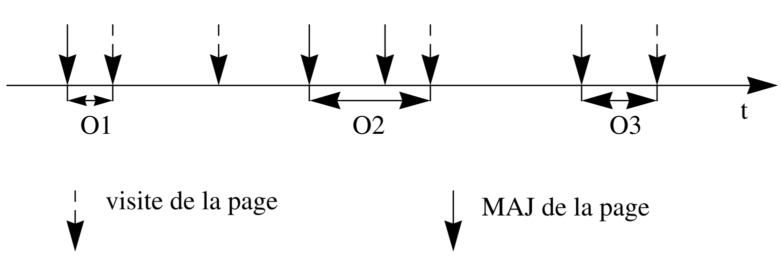
Pourtant, certains dangers ont été dénoncés par le chercheur. En effet, les développeurs de robots se sont trouvés face à face au problème de la charge des réseaux, problème majeur de l'informatique dans le futur. Alors que les navigateurs traditionnels nécessitent un travail de l'utilisateur afin de passer de page en page (opération manuelle + délai de chargement de la page), les robots font cette navigation de façon automatique. L'inconvénient principale est que cette automatisation nuit au réseau. Pour chaque nouveau lien rencontré, le robot crée une connection HTTP. La vitesse de ces opérations (requêtes d'interrogation) étant très élevée, le robot risque d'utiliser fortement la bande passante, ce qui va charger énormément le serveur HTTP et implique une saturation du réseau pour les autres utilisateurs. C'est ce qu'on a appelé les *robots impolis*. Pour les développeurs de robot, il est nécessaire qu'ils gardent une grande attention sur le parcours des pages en respectant un délai d'attente entre la visite des différentes pages. C'est pour cela que des techniques, qui limitent l'accès au serveur à certains robots, ont été créées. D'ailleurs, quelques recherches sur les robots approfondissent le *«protocole d'exclusion des robots»* (voir [9] et [10]). Il est toujours possible à un administrateur de demander à ce que son site ne soit pas visité, s'il pense que cette visite ralentit les performances de son serveur. De plus en plus de monde pense qu'il n'est pas souhaitable d'écrire de nouveaux robots pour éviter de ralentir les serveurs.

Le reste des recherches lié à *WebCrawler* est plus orienté vers le stockage des informations et la gestion des bases de données mise à jour par les robots. Nous ne nous attarderons donc pas sur cette partie.

2.4 La mise à jour des informations des bases de données du WEB

Les robots WWW ont essentiellement pour but de remplir des bases de données. Le WEB évoluant sans cesse, il est très difficile de garder ces bases mises à jour. Il faut donc le visiter de façon régulière. Le problème [6] est que, pendant un délai O_i, temps où les données sur une page sont dites hors date, la base de données contiendra des informations erronées sur la page i (voir la figure 3).

FIGURE 3. Période erronée d'une base de données



Les périodes O*n* sont les périodes pendant lesquelles une base de données contient des informations dépassées : la page a été remise à jour mais le robot n'a pas encore revisité cette page : durant le délai passé entre la mise à jour d'une page et une nouvelle visite de la page, la base de donnée contient des informations inexactes sur cette page. Il faut donc faire une analyse afin de minimiser ce délai.

La solution la plus simple serait de lancer très souvent un parcours afin de diminuer le plus possible ce temps. Mais, cette méthode risque de ralentir le réseau. Il est également important de lancer le robot le moins souvent possible. Il faut donc trouver une relation pour exécuter le programme, à dates le plus espacées, mais de façon à ce que le délai entre la visite et la modification de la page soit le plus restreint possible.

2.5 Les limites des travaux

Comme nous l'avons vu auparavant, bien que le nombre de robots existant soit très élevé, la plupart se consacre à l'indexation du WEB. Par contre, dans le domaine statistique, en vue de créer un site miroir par exemple, les recherches ne sont pas très approfondies. Le principal intérêt de copier un site miroir d'un serveur distant sur une partition locale est de diminuer les temps d'accès au serveur initial, l'accès à des pages locales étant très rapide et le coût des espaces disques ayant nettement diminué. Mais, il n'existe un intérêt à cette technique que si les deux sites, le modèle et la copie, sont identiques à n'importe quel moment. Aussi faut-il faire une sauvegarde à chaque nouvelle mise à jour. Mais, la solution de copier très souvent le site n'est pas une bonne solution puisqu'elle surcharge inutilement le réseau alors que le but premier de cette technique serait justement de diminuer l'usage de la bande passante. Il va falloir trouver un moyen de mettre à jour le site local, le moins souvent possible mais de façon à ce qu'il soit aussi le plus à jour possible. C'est ainsi que née l'idée d'analyser la mise à jour des sites WEB. Dans le cas de l'indexation, le problème de ces périodes est moins contraignant car il n'y a, en général, pas trop de changement sur le thème global d'une page.

3 Développement du robot

3.1 Le robot *Web Stator*

Dans le cadre du projet de D.E.A., le sujet est de savoir s'il existe une politique de mise à jour des pages WEB. Pour cela, le but premier est de développer un robot qui effectuerait une visite des sites WEB (comme tout moteur de recherche) et qui, pour chaque page visitée, regarderait la date de dernière modification. Ainsi, il sera possible de faire une analyse, en fonction des résultats obtenus, de la fréquence de mises à jour des pages HTML et des sites WEB en général.

3.1.1 World Wide Web Consortium

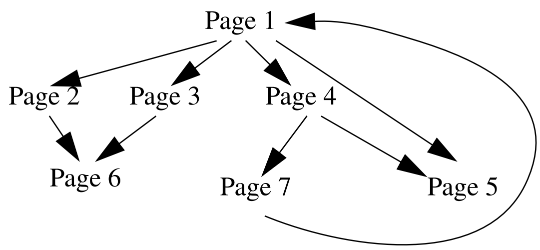
Le développement complet d'un moteur de recherche était donc la première étape du stage. Les recherches de W3 Consortium ont permis de rendre le travail plus facile. En effet, W3C [13] propose un ensemble de librairies gratuites permettant la navigation sur le Web. Ce moteur est reconnu, et ne risque pas de poser de problèmes de politesse au niveau des accès aux sites Web.

En fait, W3C offre une bibliothèque dans le langage C contenant des fonctions et des structures de travail sur le World Wide Web. Ainsi, il donne un squelette et laisse aux *adaptateurs* du robot le soin de programmer certaines fonctions qui permettent d'obtenir un travail spécifique.

3.1.2 Le parcours du robot

L'utilisateur spécifie une adresse. Le déroulement est ensuite simple. Le robot crée une connection TCP sur le serveur passé en paramètre et lui envoie le chemin de la page voulue. Ce serveur, comme dans tout protocole HTTP, retourne une chaîne de caractère, contenant des informations sur le document et les données HTML, si la page existe. Dans le cas contraire, une chaîne d'erreur (un fichier par défaut) est envoyée au client. La connection est ensuite cassée par le serveur. Une fois que le robot à récupérer les données, il va pouvoir commencer la lecture de la page. Il analyse tout le document (étape de parsing) qu'il a transféré. Dans un premier temps, il va créer des structures contenant les informations de tête du document, tel que la version du démon HTTP, la longueur du document, la date du passage ou la date de dernière modification. Ensuite, sur le corps du document (données HTML), il va créer une liste de liens référencés. A chaque fois qu'il découvre un tag de type lien (ex: *<a href= ... >*) sur une nouvelle adresse, il ajoutera l'adresse dans la liste des pages à visiter. Cette liste permettra de savoir pour chaque adresse rencontrée, s'il s'agit d'une adresse déjà apparue auparavant, ou une nouvelle à explorer. C'est en fait, une liste de pointeurs sur des structures *anchor* (structure contenant toutes les données relatives à un document : date de modification, longueur du document, ...), et la profondeur où a été visitée la page. Cette recherche va durer tant que le robot n'atteint pas la profondeur spécifiée au départ ou, tant qu'il n'a pas visité tous les liens référencés du site. On parle de profondeur de travail, pour désigner le nombre de lien que le robot doit suivre. Le site WEB peut être considéré comme un arbre parcouru en largeur. Voyons l'exemple suivant de parcours du robot.

FIGURE 4. Exemple de parcours



Initialisation de la liste de visite L = ()
^ représente le pointeur de liste
A chaque fin de visite d'une page le pointeur avancera dans la liste

Visite de la page 1 : L=(^1)
Référence 2 :L=(^1,2)
(travail identique pour 3,4,5)
Fin de la page 1 : L=(^1,2,3,4,5)

Visite de la page 2 : L=(1,^2,3,4,5)
Référence 6 : L=(1,^2,3,4,5,6)
Fin de la visite de la page 2 : L=(1,^2,3,4,5,6)

Visite de la page 3 : L=(1,2,^3,4,5,6)
Référence 6 : $6 \in L$ donc on ne fait rien
Fin de la visite de la page 3 : L=(1,2,^3,4,5,6)

Visite de la page 4 : L=(1,2,3,^4,5,6)
Référence 7 : L=(1,2,3,^4,5,6,7)
Référence 5 : $5 \in L$ donc on ne fait rien
Fin de la visite de la page 4 : L=(1,2,3,^4,5,6,7)

Visite de la page 5 : L=(1,2,3,4,^5,6,7)
Fin de la visite de la page 5 : L=(1,2,3,4,^5,6,7)

Visite de la page 6 : L=(1,2,3,4,5,^6,7)
Fin de la visite de la page 6 : L=(1,2,3,4,5,^6,7)

Visite 7 : L=(1,2,3,4,5,6,^7)
Référence 1 : $1 \in L$ donc on ne fait rien
Fin de la visite de la page 7 : L=(1,2,3,4,5,6,^7)

L=(1,2,3,4,5,6,7^)
Parcours fini : toutes les pages ont été visitées

Dans le cadre du travail demandé, il n'est pas utile de parcourir plusieurs sites puisqu'il s'agit d'analyser un site à la fois. Il serait plus intéressant de visiter entièrement toutes les pages d'un seule site et de relancer le robot aussi souvent qu'il y a de sites à étudier. Si toutes les références d'une page sont visitées, avant d'arriver au niveau n, il faudra avoir visité toutes les pages du niveau n-1. Avec une grande profondeur, le robot risquerait de se perdre dans des serveurs qui ne nous intéressent pas pour l'étude et, le délai de retour dans de tel cas risque d'être très long. Il est donc souhaitable de limiter l'accès aux pages du même serveur. Pour cela, il ne faut plus traiter, dans les tags

HTML, que les liens symboliques (chemin par rapport à la page analysée) et les liens sur le même serveur.

Le robot de W3C laisse plusieurs fonctions qui peuvent être changées selon le goût du programmeur. Ainsi, il n'est pas nécessaire de toucher à toutes les sources de la bibliothèque pour pouvoir faire des changements sur le comportement du robot. Ses fonctions sont appelées : à l'entrée et à la sortie d'une nouvelle url, à l'entrée et à la sortie d'un paragraphe, d'une ligne, d'un caractère, etc ... Les fonctions qui ont été reprises, sont HText_beginAnchor et HText_endAppend(). La première est appelée à chaque fois, lors de l'analyse d'un document, qu'un lien HTML est trouvé. C'est dans cette fonction que le robot vérifie qu'il n'a pas déjà visité cette page et si ce n'est pas le cas, il ajoute cette page dans la liste des documents à visiter. C'est également dans cette fonction que l'on vérifie si l'adresse trouvée ne sort pas du domaine et si l'on est toujours dans une profondeur désirée. La deuxième fonction est appelée lorsque le robot a analysé tout le document. A ce moment, toutes les structures sont correctement remplies et il est possible de mettre à jour les bases de données.

L'inconvénient majeur de ce robot est qu'il se limite aux tags *<a href=...>* et que les frames ne sont pas gérées. Si l'auteur d'un document souhaite découper le navigateur en plusieurs fenêtre, il doit spécifier la taille et la disposition de chaque fenêtre ainsi que le document qui sera ouvert dans chacune d'elles. Ainsi, dans la page principale, il peut n'y avoir aucun lien référencé.

Exemple de déclaration de frame

```
<FRAMESET COLS=”18%,*” BORDER=0>
  <FRAME SRC=”blanc.htm”
   MARGINHEIGHT=0
   MARGINWIDTH=0
   NAME=”blanc”
   SCROLLING = ”auto”>

<FRAMESET ROWS=”82%”>
  <FRAME SRC=”homepage.html”
   MARGINHEIGHT=0
   MARGINWIDTH=0
   NAME=”homepage”
   SCROLLING = “auto”>
 </FRAMESET>
```

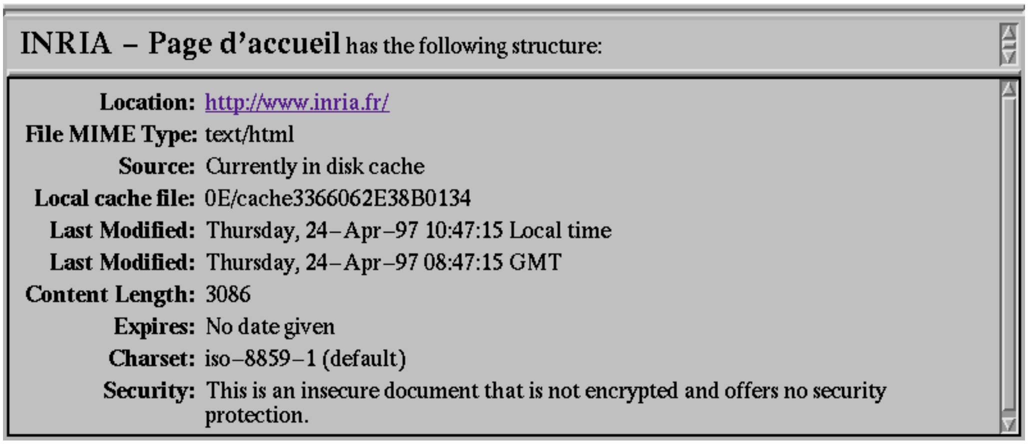
En fait, les références sont faites par les documents de chaque frame. Actuellement, le robot ne traite pas de tel cas de page. Pour pouvoir étudier, des sites comprenant plusieurs frames, le robot sera lancé sur la page paraissant la plus importante. Il serait possible de reprendre l'analyse de la page pour rajouter le tag frameset.

3.1.3 Le travail particulier de Web Stator

En fait, il faut maintenant adapté ce robot pour rapatrier toutes les données nécessaires à l'analyse. Cette solution a été considérée comme peu réalisable. Pour pouvoir tirer des hypothèses sur les résultats, il faudrait déjà faire plusieurs parcours d'un même site et suivre les mêmes liens d'une visite sur l'autre. Le problème d'utiliser toujours le même robot pour ce parcours risque de poser le problème suivant. Si entre deux visites, des pages sont supprimées ou ajoutées, le parcours sera complètement différents d'un cas sur l'autre. Et l'analyse s'en trouvera faussée. Aussi, l'utilisation de deux robots est une meilleure solution. Le premier, dérivé de celui de W3C, fera un parcours du site afin d'en déterminer la hiérarchie globale du serveur étudié. Une fois, cette visite effectuée, il mettrait à jour une base de données qui serait réutilisée pour la suite des recherches. Le deuxième robot sera lancé sur les résultats du premier, de façon à pouvoir suivre un processus Poisson entre les différentes visites d'une même page. Nous verrons dans le chapitre 3.2. pourquoi et comment utiliser ce processus Poisson.

Le premier robot va être lancé sur une URL et parcourir récursivement tout le site donné. Pour chaque adresse, il va stocker l'adresse courante, l'adresse qui a fait référence à la page courante (NULL pour le premier document) et la date de passage. Le deuxième robot va utiliser les URL rapatriées pour les utiliser comme chemin à suivre. Dans ce cas, le robot ne va pas du tout regarder les liens contenus dans les documents visités. Il va interroger la base de données dans un premier temps pour connaître le chemin qu'il doit suivre. Une fois cette liste obtenue, pour chaque URL trouvée, il va ramener les informations liées au document par le protocole HTTP décrit précédemment. Ayant ainsi récupéré ainsi une chaîne de caractère, il va déterminer la date de dernière modification du document en question et la stocker dans la table avec la date de passage du robot. Les données rapatriées par le robot peuvent être trouvées de façon manuelle avec un navigateur tel *netscape*, en utilisant Document Info dans le menu View (voir figure 5).

FIGURE 5. Fenêtre ***Document info*** de Netscape



Le problème posé est que, les données telles que la date de mise à jour, la taille du document, dépendent de la configuration du démon HTTP qui regarde la dernière modification du fichier, sa taille, ... Aussi, le robot lancé sur certains serveurs WEB mal configurés, est incapable de récupérer ces dates de mises à jour. De plus, le format des dates rendu n'est pas toujours le même. Pour la suite du projet, la plupart des tests seront faits sur le serveur de l'I.N.R.I.A. pour des raisons pratiques de performance. En effet, c'est un gros serveur assez diversifié, sur lequel il est possible de faire des statistiques, car il est composé de pages personnelles, de pages de présentation du serveur, de pages d'informations ... De plus, c'est un serveur local, ce qui diminue les délais de visite.

3.1.4 La base de données

Dans la suite du stage, le but est de faire une analyse statistique des résultats obtenus par le robot. Pour cela, la sauvegarde des résultats a dû se faire de façon à pouvoir réutiliser facilement ces données stockées. La première idée d'utiliser un fichier pour stocker le résultat de l'exécution n'étant pas une bonne chose, puisque l'accès ensuite aux informations n'est pas pratique, l'utilisation d'une Base de Données s'est avérée une bien meilleure solution. Les informations pourront être ainsi retrouvées beaucoup plus vite, grâce à un langage de requêtes, que par l'intermédiaire d'un fichier qui nécessite un parcours séquentiel.

Aucune Base de Données n'étant disponible dans le projet MISTRAL de l'I.N.R.I.A, il a fallu en trouver une. Le produit ***msql de Hughes Technologies*** [8] a été choisi pour différentes raisons. En effet, c'est une véritable base de données, avec un langage d'interrogation se rapprochant beaucoup de SQL (avec quelques fonctionnalités en moins), et de plus surtout gratuit. La sauvegarde des résultats et la recherche des informations se fera ainsi bien plus facilement. De plus, msql offre une bibliothèque de fonctions C de dialogue avec la Base (connection, interrogation, MAJ, ...). Le robot va donc se servir de cette bibliothèque pour mettre la base à jour.

3.1.5 Le stockage des données

Msql n'étant pas une base de données très puissante, le développement du robot a dû tenir compte des délais de réponse. Pour éviter trop d'accès concurrents, qui ralentissent le système, toutes les mises à jour du deuxième robot se feront de manière différée. Car, les requêtes qui étaient faites de manière régulière ont posé des problèmes quand la base a commencé à contenir un grand nombre d'enregistrement. Dans le cas du premier robot, seul une liste d'adresses a besoin d'être stockée pour conserver le chemin suivi. Dans le cas du deuxième robot, il faut non seulement, interroger la base pour récupérer ce chemin, mais aussi garder les informations sur les adresses, les dates de modification du document et de passage du robot. De plus, il faudra également stocker le temps de décalage entre chaque visite : ce temps est nécessaire au processus Poisson.

Le stockage des dates va se faire ainsi : année/mois/jour-heure:minute:seconde, l'année étant codée sur quatre chiffres pour éviter les problèmes de fin de siècle. Ce genre de formatage de dates est beaucoup plus facile à utiliser (surtout au niveau des tris) que les dates standards du type *«Thu, 13 Mar 1997 11:16:18 GMT»* que le robot est capable de rapatrier dans un premier temps. Il faut donc coder les dates obtenues. De plus, il fallait garder également une précision de l'ordre de la seconde (une page peut être modifiée plusieurs fois dans la même journée). Pour pouvoir faire des différence entre deux dates, la solution utilisée a été de ramener chaque date en seconde par rapport à une date initiale. Cette date choisie est le 1er janvier 1970. Il suffit ensuite de soustraire les deux montants obtenus et d'obtenir un résultat en seconde.

3.1.6 Les documents à visiter

Le robot est capable de visiter tous les types de documents (texte, image, son,...), mais seuls certains ont une valeur pour la suite des recherches. Toutes les pages du type image ne sont pas accéder car elles n'évoluent pas dans le temps. Par contre, si une image change dans une page HTML alors c'est la date de mise à jour de cette page qui va changer. Donc, seules les pages de textes sont intéressantes à visiter. Comme nous l'avons vu auparavant, le robot s'est limité à la visite des pages sur un même serveur. Il faut également qu'il se limite à certaines extensions : seules les pages .html (ou .HTML) (format standard), .htm (extension PC), ou bien les répertoires (puisque par défaut ce sont les fichiers index.html ou homepage.html qui sont ouverts) seront visitées.

Une question s'est posée lors de la restriction des accès sur certaines pages : faut-il visiter les pages générées par un programme cgi? Un programme cgi est un programme (écrit dans n'importe quel langage de programmation) qui fait des écritures sur la sortie standard. Exécuté par l'intermédiaire d'un navigateur, le programme, avec extension spécifique (.cgi, .pl, ...), écrit dans un premier temps une ligne où il indique qu'il va renvoyer du texte HTML et puis renvoie ensuite le texte qui sera perçu directement comme une page html. Le problème des programmes cgi est de savoir si leur date de modification est la date à laquelle a été développé le programme, ou bien est-ce à la date où a été exécuté ce programme. En fait, la deuxième solution s'est avérée plus juste puisque un programme cgi peut ne jamais donner le même résultat. En effet, suivant l'utilisateur qui lance le programme, ce cgi peut indiquer le nom, prénom, adresse de l'utilisateur. Ces informations changeront à chaque nouvel utilisateur. Dans le cas du développement d'un site miroir, il ne faudrait pas rapatrier ce type de page car son exécution pourrait varier si le programme fait accès à une base de données, mise uniquement à jour sur le site distant.

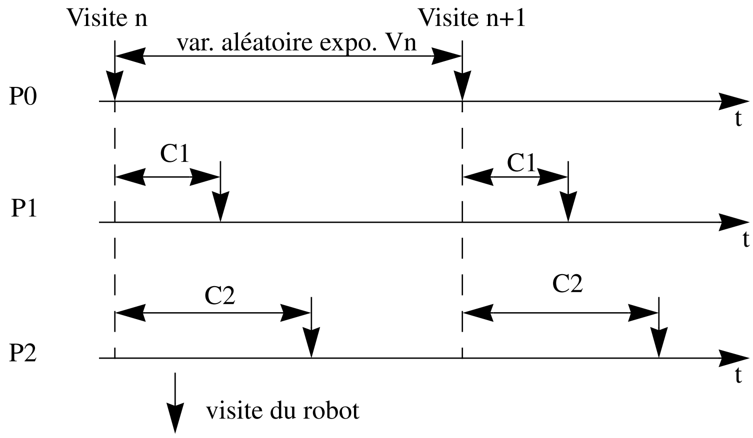
Donc, la visite des programmes cgi semble être inutile puisque l'on connaît déjà, à n'importe quelle date, sa dernière mise à jour, puisqu'il s'avère que c'est la date de visite de la page en question (exécution du programme). Il ne serait alors pas très intéressant d'utiliser ce type de pages pour faire des statistiques sur la mise à jour des sites WEB.

3.2 Le processus Poisson

3.2.1 Les visites de pages

Pour obtenir un certain type de mesure, le robot qui rapatrie les dates de modifications, soit suivre un processus de Poisson dans sa visite de page pour permettre l'analyse des résultats. C'est à dire que les moments d'observation doivent être pris de façon totalement aléatoire. Ainsi, il sera possible de tirer des conclusions sur les espérances des mises à jour des sites à partir des espérances obtenues entre la visite et la date de dernière modification de chaque page. Pour cela, nous allons procéder de la manière suivante (figure 6).

FIGURE 6. Chronologie des visites d'un site

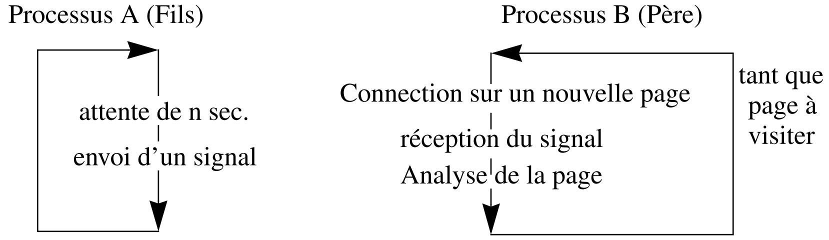


Il est important que la visite de chaque page suive elle-même un processus Poisson. Pour cela, l'intervalle entre les visites d'une même page doit être déterminée dans le temps par une variable aléatoire exponentielle. Nous verrons dans la section suivante comment déterminer cette variable. Nous voyons également pourquoi il est important de garder le même chemin de parcours d'une visite sur l'autre.

Pour se faire, il faut «ralentir» le robot afin qu'il consulte une page au moment voulu. C'est à dire qu'il faut attendre un certain délai entre la visite de chaque page. Ce délai sera identique entre les différentes visites d'un même site. Il est représenté dans le schéma ci-dessus par les constantes Cn.

Le délai C1 sera le temps mis par le robot pour passer d'une page P0 à une page P1. Mais ce délai n'est pas uniquement une mise en sommeil du robot. En fait, il faut considérer le temps réseau nécessaire au passage d'une page à l'autre. Il faudra donc calculer le temps d'attente effectif. Pour déterminer les constantes Cn, il est impératif que ces délais soient au moins supérieurs au temps déterminé par le réseau. les Cn seront d'autant plus grands que le robot tourne sur un site distant ou sur un réseau saturé. Pour mettre au point la stratégie d'attente, l'idée est d'utiliser deux processus comme dans le schéma suivant.

FIGURE 7. Synchronisation des visites entre plusieurs pages



En fait, il s'agit de créer deux processus Père et Fils, le premier envoyant tout simplement un signal au deuxième toutes les périodes données. Le deuxième fait la majeure partie du travail (transfert des données d'un document), et se bloque entre les visites de deux pages jusqu'à l'obtention du signal. Cette méthode permet de ne pas s'intéresser au temps réseau pour obtenir la connection sur la page. Il faut juste s'assurer que le temps de repos du fils est au moins supérieur au temps écoulé pour la connection sur une page.

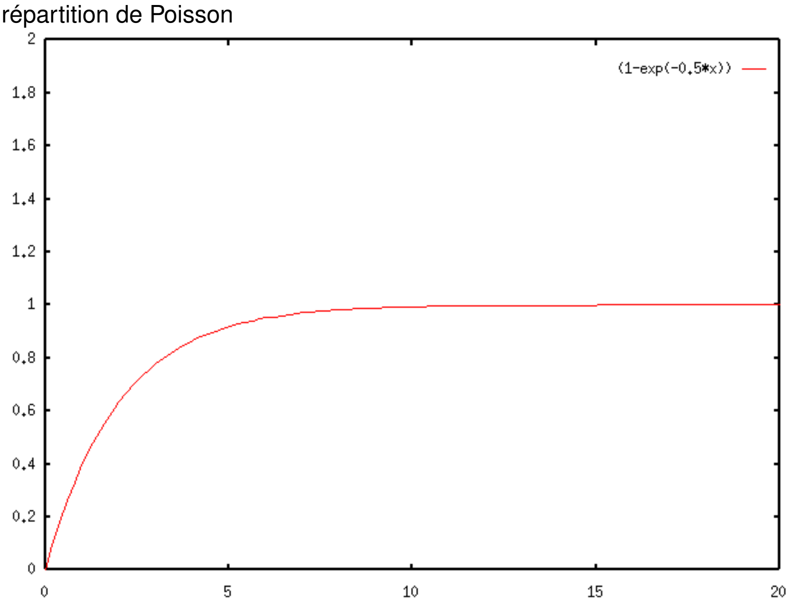
3.2.2 Comment déterminer le délai entre les visites

Comme vu précédemment, il faut qu'une page soit visitée suivant un processus de Poisson. Pour cela, il faut que la période entre les visites soient déterminées de façon totalement aléatoire.

On dit qu'un processus suit une loi de Poisson s'il se déroule à des périodes de temps aléatoires exponentielles. La fonction C, *rand()*, est une fonction qui renvoie des entiers. En fait, elle renvoie des valeurs uniformément réparties sur un intervalle (de 0 à $2^{31}-1$). Pour pouvoir poursuivre, il faut ramener ces variables sur un intervalle de $[0,1]$ (division par la borne supérieure par exemple). Il faut ensuite se ramener à des variables aléatoires exponentielles, utiles pour la suite. Pour passer de variables aléatoires uniformément réparties à des variables aléatoires exponentielles, il faut utiliser la fonction inverse de la fonction de répartition de Poisson.

Soit le fonction de répartition de Poisson F, tel que : $F(x) = P(X \le x) = 1 - e^{-\lambda x}$

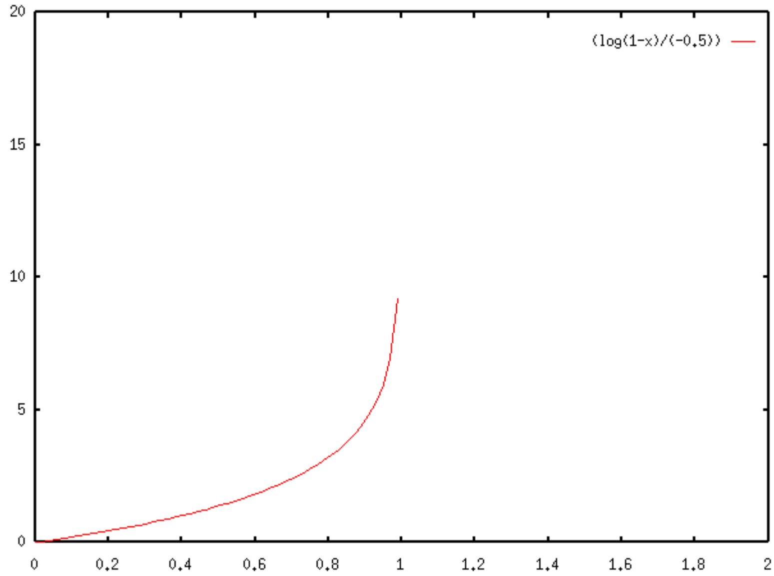
FIGURE 8. Fonction de répartition de Poisson



Supposons la fonction inverse de F, F^{-1} on sait que : $F(x) = 1 - e^{-\lambda x} = y$

donc : $F^{-1}(y) = \frac{\log(1-y)}{-\lambda} = x$ avec $y \in [-\infty,1]$ et $\lambda \ne 0$ et où y est une variable aléatoire uniformément répartie sur [0;1].

FIGURE 9. Fonction inverse de Poisson



Le problème principal de ce calcul est le problème de précision des machines. En effet, nous voyons que $\lim_{y \to 1} F^{-1}(y) = \lim_{y \to 1} \frac{\log(1-y)}{-\lambda} = \infty$ alors que la précision lors de la programmation est beaucoup moins bonne qu'en théorie.

Pour changer d'échelle, suivant la distance du site, il est possible de faire varier la valeur de λ afin d'adapter les valeurs des variables aléatoires exponentielles.

4 Les résultats

Pour la suite de l'analyse, les mesures se sont restreintes uniquement à certains serveurs. Nous allons étudier les sites suivants : www.inria.fr (serveur local de l'inria), www.inra.fr (serveur de l'institut national de recherche agronomique), www.edf.fr (Electricité de France), des serveurs de télévision et de journaux. Bien sûr, le nombre de pages constituant un site ne est donné qu'à titre indicatif, et peut changer d'un jour sur l'autre.

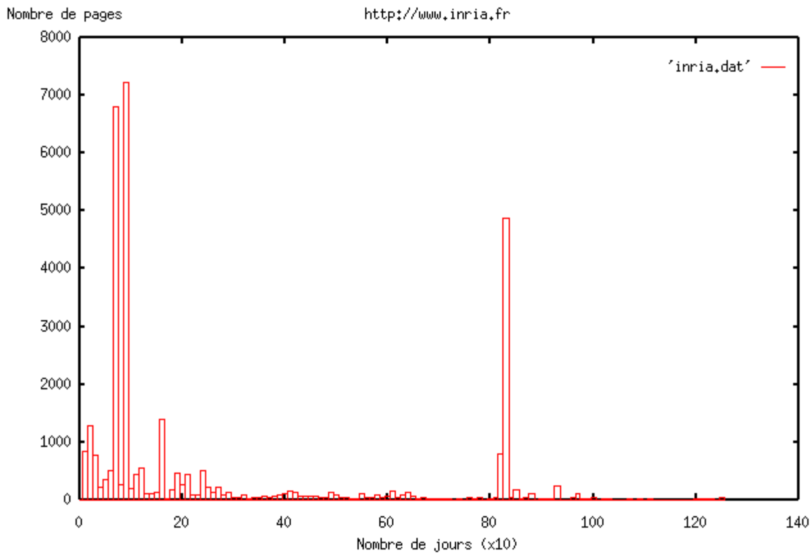
4.1 Des serveurs de laboratoires de recherche

Les laboratoires de recherches créent des sites Web pour présenter leurs travaux. Souvent, ces serveurs sont composés de différents projets. On peut également distinguer des pages de présentation des lieux et travaux, des rapports de recherches et d'activités, des pages personnelles. Il existe donc plusieurs groupes de pages qu'il est possible de distinguer par rapport à leur mise à jour.

4.1.1 Institut National de Recherche en Informatique et Automatique

Sur le serveur local, nous pouvons voir que la moyenne de mise à jour est très élevée, soit environ une moyenne de 271 jours, ce qui signifierait que les pages de l'I.N.R.I.A. sont modifiées seulement tous les neufs mois. La fréquence de mise à jour étant si peu élevée, il serait facile de conclure que le site est très stable et que peu de changement ne sont faits. Pourtant, en regardant un peu l'histogramme suivant, nous allons découvrir de nouvelles données.

FIGURE 10. Histogramme de fréquences du serveur de l'I.N.R.I.A.

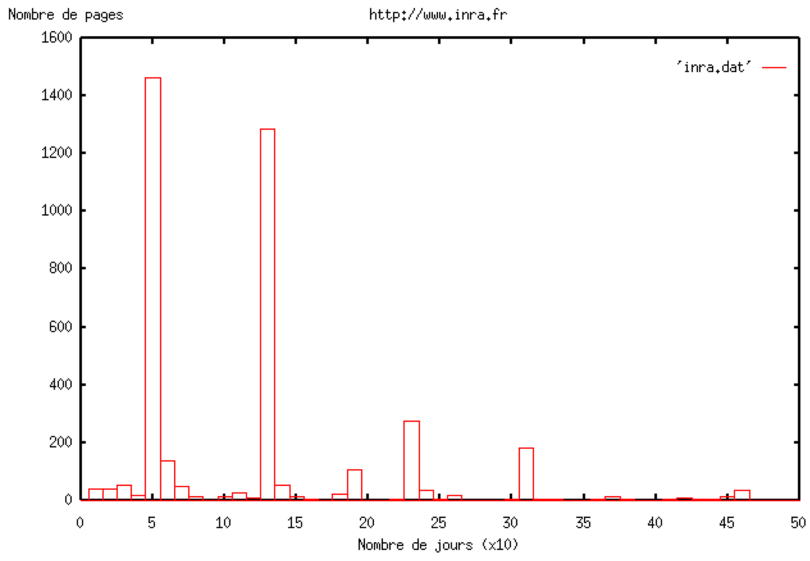


Le serveur est constitué d'à peu près 16640 pages. Les premiers changements constatés les premiers mois ne sont pas très significatifs car beaucoup de pages, d'habitude stable, ont été changées pour améliorer la convivialité du site. Par contre une grosse activité est à signaler lors du deuxième mois. Aussi, nous pouvons déduire que l'hypothèse comme quoi les pages de ce site ne sont pas très modifiées est fausse. Alors comment expliquer une telle fréquence? Si nous nous attardons quelques seconde de plus sur ce schéma, nous voyons de forts changements, il y a environ 17 mois. On peut considérer ces pages comme «*mortes*» car il y a peu de chances qu'elles évoluent encore dans le temps. Si elles n'ont pas bougé, il est possible de l'expliquer : ce sont des rapports de travaux qui, une fois faits, n'évoluent guère. Dans le cas d'une création d'un miroir il ne serait pas très intéressant de visiter ces pages très souvent, car, à part cas exceptionnel, il n'est peu probable que ces pages évoluent encore. Par contre, les autres pages doivent être plus souvent visité afin de voir leur évolution dans le temps.

4.1.2 Institut National de Recherche Agronomique

Le serveur de l'I.N.R.A. (*http://www.inra.fr*) est un serveur de 3920 pages différentes. L'espérance de ce site est de 2718 heures soit environ 113 jours. C'est à dire que l'on peut donc considérer que ce site a une moyenne de mise à jour s'échelonnant dans une période d'environ quatre mois. Cette moyenne étant tout de même assez grande, il serait intéressant de voir comment se comportent les différentes pages du site.

FIGURE 11. Histogramme de fréquences des pages de l'I.N.R.A.



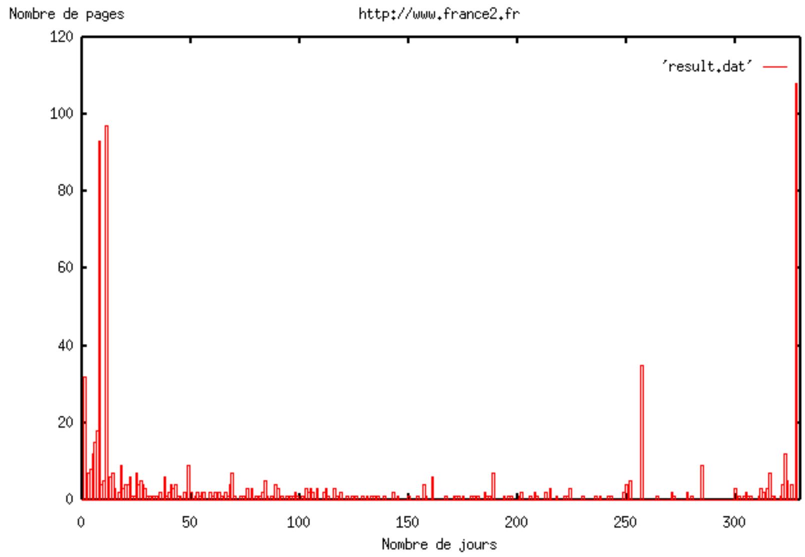
On peut distinguer deux modes dans le graphe ci-dessus. Il s'agit en fait d'une forte mise à jour des pages constituant le site. Ils apparaissent à 50 et 130 jours de la date de visite. On peut également voir que seul, 3% des pages du sites ont été changées dans le dernier mois. On peut donc considérer que ce site a une activité moyenne du point de vue des mises à jours.

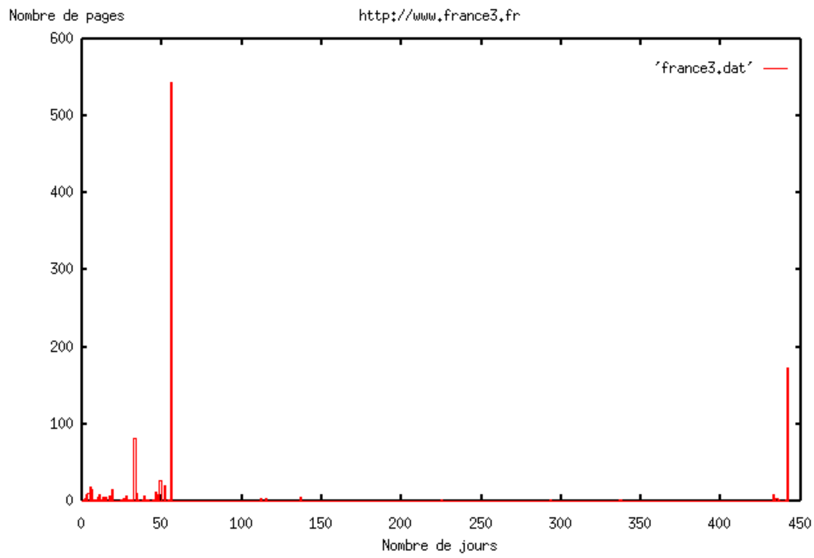
4.2 Le web de France Télévision

Si les serveurs des télévisions ont été choisis, c'est pour étudier le comportement d'un site dont les mises à jour doivent être fréquentes. En effet, il est probable que certaines pages soient modifiées régulièrement du moins celles donnant le programme de la semaine.

Il existe deux serveurs pour France Télévision, d'une taille de 808 adresses pour France 2 et de 1027 adresses pour France 3. En comparant l'espérance de ces deux sites (117 jours pour france 2 et 120 jours pour france 3), il semblerait que la mise à jour de ces serveurs est à peu près identique. Pourtant, les histogrammes suivants vont nous apporter plus d'informations.

FIGURE 12. Histogrammes des fréquences des serveurs de France Télévision



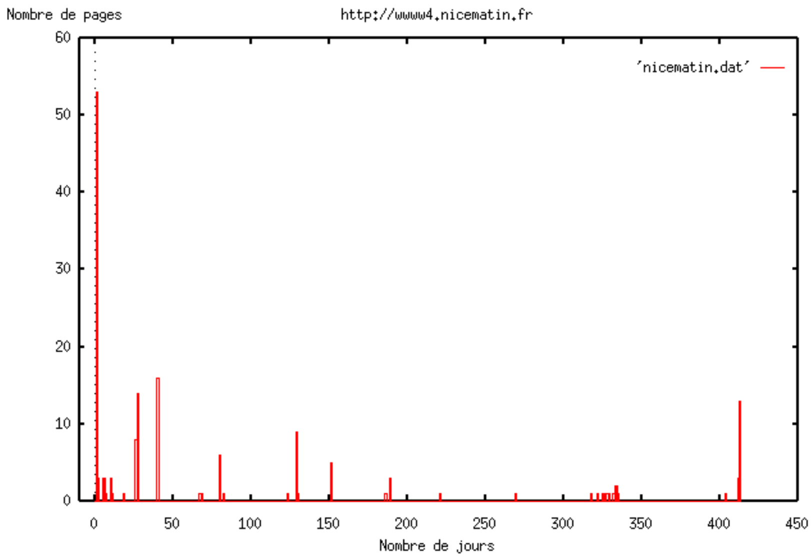


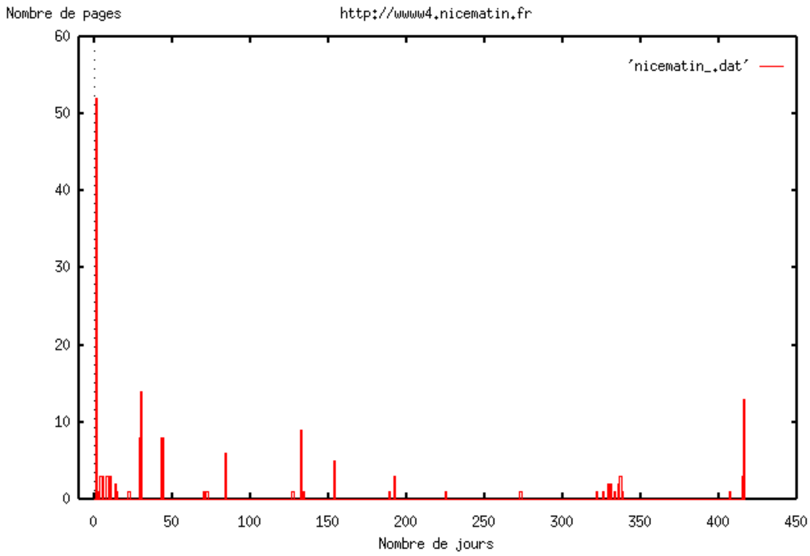
Avec ces histogrammes, il est facile de voir que ces deux serveurs n'ont pas du tout le même comportement. Un point commun, cependant, est à signaler : ces deux serveurs ont dû être créés dans les mêmes périodes. A peu près le même nombre de page reste inchangé depuis dizaine de mois. Il s'agit des pages de présentation qui n'ont pas évolué depuis. Par contre, sur le serveur de France 3, nous pouvons distinguer qu'aucune (ou presque aucune) page n'a été changée durant la période de 430 jours à 50 jours avant la date de visite du robot. Deux explications sont possibles. Soit un certain nombre de page ont été créée à la date j-50 (mode de l'histogramme), soit toutes les pages ont été modifiées. Une étude plus approfondies dans le temps pourrait distinguer ces deux cas. Le site de France 2 a tout de même une plus grande activité que celui de France 3.

4.3 Le serveur de Nice Matin

L'étude du serveur de Nice Matin permet de suivre l'évolution d'un serveur mis à jour quotidiennement. Il a pourtant une fréquence moyenne de mise à jour de 100 jours qui ne reflète pas ces changements quotidiens. L'explication tient dans le fait que beaucoup de pages n'ont pas été changées depuis la création du service. En fait, ces pages resteront stable jusqu'à ce qu'il y ait un changement global de la présentation du site. Par contre, voyons deux histogrammes de fréquences à une semaine d'intervalle.

FIGURE 13. Histogrammes de fréquences du serveur de Nice Matin à deux dates différentes





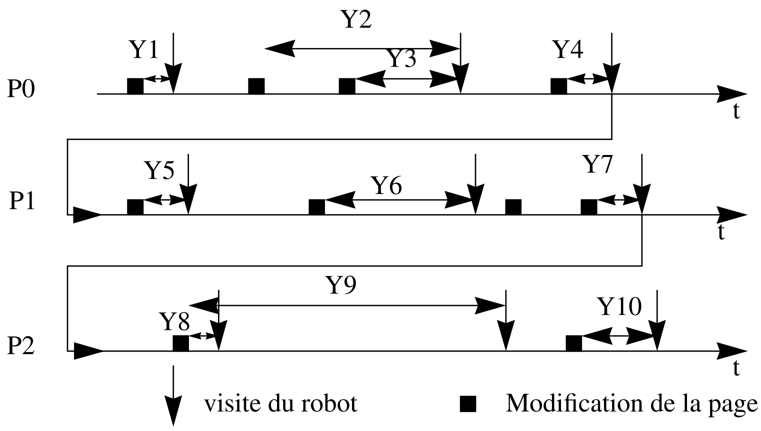
Le mode trouvé est identique à plusieurs dates différentes. Cela signifie qu'un tiers des pages sont renouvelées de façon quotidienne. La moyenne n'est donc pas très représentative de l'ensemble du site. Elle va sûrement continuer d'augmenter avec le temps car les pages qui n'ont pas été modifiées depuis longtemps risque de ne plus «jamais» évoluer. Par contre, la politique de mise à jour devrait être stable du point de vue des autres pages. Aussi, est-il important de ne pas trop se fier à l'espérance d'un site.

4.4 L'évolution des sites

Pour pouvoir approfondir les statistiques, multiplier les mesures serait nécessaire. Nous allons revenir sur le processus Poisson vu plus tôt. Considérons un site de N pages. Un premier robot va le parcourir pour en déterminer la hiérarchie générale. Une fois cette ordonnancement obtenu, prenons un second robot, qui va suivre cette hiérarchie suivant un processus de Poisson afin de rapatrier les dates de dernière modification des différentes pages visitées.

Ne considérons plus, à partir de maintenant, le domaine comme un site de N pages mais comme une grande page constituée de ces N pages initiales. Cette concaténation permet d'obtenir une page unique souvent visitée. On multiplie ainsi le nombre de visite sans pour autant exécuter plus souvent le robot.

FIGURE 14. Concaténation des pages d'un même site



La page concaténée est composée des N pages : la page 0 visitée entre t_0 et t_1, la page 1 entre t_1 et t_2, etc. Ainsi si V visites ont été programmées, on pourra considérer que la page ainsi créée aura été visitée V x N fois. C'est ainsi, qu'il est possible de faire des statistiques avec un faible échantillon représentatif.

Il faut que les points d'observation suivent un processus de Poisson, c'est à dire que les dates des visites soient obtenues de façon aléatoire. Considérons que nous obtenons des variables Yi représentant la durée écoulée entre la dernière modification et la visite d'une page. Supposons les variables Xi représentent la période écoulée entre deux modifications. Nous avons alors :

$E(y) = \int_0^{\infty} yf(y)dy$ où f est la fonction de densité.

Il est plus simple de se servir de l'égalité obtenue par la loi *limite centrale* qui nous donne un résultat basé sur les moyennes des variables Yi.

$E(y) = \lim_{n \to \infty} \frac{1}{n} \sum_1^{\infty} Yi$. Les variables Yi étant des variables indépendante uniformément distribuée.

Il est donc possible de calculer l'espérance des visites d'un site à partir des résultats rapatriés par le robot. De plus, dans le cas général, on peut en déduire : $E(Y) = \frac{E(X^2)}{2E(X)}$. car nous sommes dans un cas de processus de Poisson pour les variables Y.

Si l'on suppose que les mises à jour des pages suivent également un processus Poisson, et grâce à la théorie de renouvellement, on sait que $E(X^2) = 2E(X)^2$. Il est ainsi possible de réduire l'équation précédente en : E(Y)=E(X).

En restant dans cette supposition que E(X)=E(Y), il est simple calculer l'espérance des variables Y, qui représente le temps écoulés entre la modification de la page et l'observation de cette même page par le robot. Aussi, nous en déduisons par conséquent l'espérance des modifications du site.

4.5 Les limites

Pourtant, comme vu dans les exemples précédents, ce calcul de l'espérance n'est pas très significatif. Un serveur Web n'est que très rarement homogène et pour obtenir une espérance qui relate la réalité, la solution est de décomposer le site en plusieurs groupes représentatifs. Cette décomposition pourrait se faire sur les modes des histogrammes de fréquences. Ensuite, il sera possible d'analyser les groupes obtenus et d'en tirer des conclusions car les échantillons seront beaucoup plus représentatifs et les résultats plus justes.

Il serait également possible de faire cette étude en séparant chaque page, mais il faudrait élargir la durée d'observation d'autant plus, ce qui parait peu réalisable. La meilleure solution reste la décomposition d'un site Web par catégorie représentative.

5 Conclusion

Le déroulement global de ce stage peut être décomposé en trois parties. La première a permis d'étudier ce qui avait été déjà réalisée dans le domaine des moteurs de recherche d'informations aussi bien au niveau indexation que statistique.

La deuxième partie de ce stage a été la phase de développement où nous avons proposé un robot répondant à un certain nombre de caractéristiques afin d'être utilisé pour l'analyse de la mise à jour du World Wide Web. Ce délai s'est avéré plus long que prévu et a, de ce fait, empiété sur la phase statistique. Aussi, l'exécution du robot s'est faite sur une échelle de temps trop restreinte, et l'analyse des résultats n'a pas vraiment reflété le comportement réel des sites étudiés.

Une étude sur un échantillon plus grand de sites et sur une période plus longue permettrait d'obtenir des données beaucoup plus significatives sur la politique de modification des pages. Il serait ainsi possible de décomposer les sites en groupes homogènes et de regrouper plusieurs sites ayant des comportements semblables. Il faudrait donc multiplier les visites du robot sur les serveurs choisis pour ne pas uniquement regarder les fréquences de mise à jour à un moment donné mais dans un intervalle de temps donné.

De nombreuses perspectives peuvent être approfondies. Si une vraie politique de mise à jour est établie, il serait ensuite possible de mettre au point la construction d'un site miroir. La copie se ferait en fonction des données obtenues par la première étude. Ces résultats détermineraient les dates de changements du site distant et donc le moment où doit être réalisé le rapatriement global du site distant sur une partition locale.

6 Bibliographie

6.1 Référence papier

[1] Ian Cooper, Laboratoire Informatique de l'Université du Kent. *Indexing the WORLD*, Novembre 1994.

[2] Murray R. Spiegel, *Theory and Problems of STATISTICS*, McGraw-Hill, 1988.

[3] Simon E. Spero, *Analysis of HTTP Performance problems,*

[4] Paul De Bra, Geert-Jan Houben, Yoram Kornatzky. *Navigationnal Search in the World-Wide Web.*

[5] Keith D. Fischer. *World Wide Web robots and spiders.* Novembre 1995.

[6] E. G. Coffmann, Jr., Zhen Liu, Richard R. Weber. *Optimal Robot Scheduling for Web Search Engines.* Janvier 1997.

6.2 Référence Web

[7] Peregrinator : *A Web-Indexing Robot*.
http://www.maths.usyd.edu.au:8000/jimr/pe/Peregrinator.html

[8] Hughes Technologies. *The msql library.*
http://www.Hughes.com.au/library/msql2/manual/

[9] Martjin Koster. *A Standard for Robot Exclusion.*
http://info.webcrawler.com/mak/projects/robots/norobots.html

[10] Martjin Koster. *Robots exclusion.*
http://info.webcrawler.com/mak/projects/robots/exclusion.html

[11] Martjin Koster. *Guidelines for Robot Writers.*
http://info.webcrawler.com/mak/projects/robots/guidelines.html

[12] Michael J. Hannah. *HTML Reference Manual.*
http://www.sandia.gov/sci_compute/html_ref.html

[13] Henrik Frystyk Nielsen. *Libwww - the W3C Sample Code Library.*
http://www.w3.org/pub/WWW/Library/