

PROXYS INTERNET AVANCÉS

THÈSE N° 2762 (2003)

PRÉSENTÉE À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

SECTION D'INFORMATIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Philippe ROCHAT

ingénieur informaticien diplômé EPF
de nationalités suisse et italienne, originaire de L'Abbaye et Le Lieu (VD)

acceptée sur proposition du jury:

Prof. S. Spaccapietra, directeur de thèse
Prof. S. Ghernaoui Helie, rapporteur
Prof. A. Schiper, rapporteur
Prof. K. Yétongnon, rapporteur

Lausanne, EPFL
2003

à Sabina

"[...] une pensée est la conséquence inévitable de la précédente et dans le cas où une pensée en engendrerait plus d'une autre (disons deux ou trois, équivalentes quant à toutes leurs implications), il sera non seulement nécessaire de suivre la première jusqu'à sa conclusion mais aussi de revenir sur ses pas jusqu'à son point d'origine, de manière à reprendre la deuxième de bout en bout, puis la troisième, et ainsi de suite, et si on devait essayer de se figurer mentalement l'image de ce processus on verrait apparaître un réseau de sentiers, [...] telle une carte"

Paul Auster
L'invention de la solitude

"Le facteur tient dans sa sacoche l'écheveau des fils qui relie entre eux les humains."

Erik Orsenna
Deux étés

REMERCIEMENTS

L'auteur voudrait avant tout remercier son directeur de thèse, Monsieur Stefano Spaccapietra, pour sa confiance, son soutien et pour avoir toujours accueilli avec enthousiasme les propositions les plus originales.

Nos remerciements vont également à Monsieur R.-D. Hersch, Professeur à l'Ecole Polytechnique Fédérale de Lausanne qui nous a fait l'honneur de présider le jury de cette thèse, à Madame Solange Ghernaouti Hélie, Professeur à l'Université de Lausanne, Monsieur Kokou Yetongnon, Professeur à l'Université de Bourgogne, Monsieur André Schiper, Professeur à l'Ecole Polytechnique Fédérale de Lausanne pour avoir accepté d'être les rapporteurs de ce travail.

Ce travail n'aurait jamais vu le jour sans le soutien constant et l'attention de Sophie Monties, qui a non seulement apporté les conseils les plus judicieux mais trouvé les bonnes issues dans les situations désespérées.

De nombreuses personnes parmi les collaborateurs du Laboratoire de Bases de Données ont également contribué à la réussite de ce travail par leurs remarques ou leur intérêt dans un domaine qui ne leur était pas toujours familier. En particulier: Abdel, Christelle, Christine, Paul, Pier et Yves qui n'auront aucune peine à reconnaître la pierre qu'ils ont apportée à l'édifice.

Un grand merci également à toutes les personnes qui, par leur relecture, ont contribué à améliorer la qualité du texte et plus particulièrement Valérie et John ainsi que Ramouna pour ses conseils d'helléniste distinguée.

L'auteur tient également à remercier tous ses amis rencontrés dans différentes aventures comme Balélec, Fréquence Banane ou encore Satellite et sans qui la vie ne serait qu'un désert.

Un grand merci aussi à sa famille, ses parents et son frère qui ont toujours eu foi en lui.

Enfin, aucun mot ne pourra décrire la gratitude ressentie pour Sabina, tendre épouse dans les épreuves les plus difficiles.

RÉSUMÉ

Depuis le début des années 90, l'Internet a énormément évolué aussi bien en nombre qu'en diversité de services disponibles. Dans cette évolution, le proxy est un acteur important et très largement répandu. A l'heure actuelle, les proxys n'ont pour fonction que d'accélérer l'accès aux contenus par le biais du caching et se basent uniquement sur des statistiques qui transparaissent dans la fréquentation d'un groupe d'utilisateurs indistincts. Les évolutions dans ce domaine se dirigent soit vers des infrastructures marginales en sacrifiant l'universalité du Web, soit vers des applications individuelles (filtrage). Nous proposons de faire évoluer les proxys vers une plateforme intermédiaire ouverte avec des fonctionnalités de caching avancées, tout en maintenant un très fort niveau de compatibilité avec les paradigmes existants. Nous basons nos développements sur deux qualités déjà présentes dans tous les proxys Internet, mais peu exploitées: leur situation privilégiée (plésiocentrique) sur les infrastructures réseau et leur perceptivité que nous étendons à de nouvelles dimensions, notamment sémantiques (engnose).

Dans un premier temps, nous élaborons une nouvelle indexation des ressources en associant aux documents un espace, que nous basons sur une position absolue extraite de la sémantique des URL, et une position relative, définie à partir des liens entre les ressources. Nous étendons les statistiques d'utilisation à ces nouvelles dimensions pour constituer un espace topologique prenant en compte la situation des documents et la navigation des utilisateurs dans cet espace. Nous proposons des algorithmes et des fonctionnalités pour construire, maintenir et exploiter cette topologie. Dans une optique d'accélération, cet espace statistique nous permet de mettre en place un système de prefetching basé sur des chaînes de Markov.

Afin d'étendre la perceptivité du proxy aux utilisateurs en amont de la chaîne de consommation, nous proposons un mécanisme de maintien de sessions basé sur des proxy-cookies. Pour cela, nous élaborons deux nouvelles directives HTTP similaires à celles définies pour les cookies. Ce paradigme nous permet également de mettre en place des services personnalisés en supportant le concept d'interaction et de profil utilisateur. Ce dernier point débouche notamment sur des problèmes liés à la mobilité et apporte une indépendance de déploiement par rapport à la disposition des infrastructures au niveau physique.

Dans un deuxième temps, nous étudions la prise en compte possible par les proxys de nouvelles dimensions dans le cadre du Web sémantique et des ontologies. Les nouvelles technologies qui entrent dans ce contexte, en particulier XML et les annotations, apportent de nouvelles informations. Ces dernières sont facilement exploitables par des processus informatiques. Quant aux annotations, elles enrichissent considérablement les informations perceptibles par le proxy, notamment par le biais de classifications dans des ontologies. Nous soulignons l'opportunité pour les proxys, de par leur situation intermédiaire et plésiocentrique, d'intégrer les fonctions de serveur d'annotations. Nous démontrons l'acquisition qui peut ainsi être faite et les avantages à en retirer par la définition de l'engnose. Il s'agit d'une nouvelle qualité du proxy qui le rend perceptif aux connaissances disséminées sur le Web. Nous proposons ainsi une nouvelle gestion du cache par métier basée sur une technique de cache virtuel à étages (partition verticale et horizontale). Nous présentons un algorithme de bascule automatique dans le domaine courant de l'utilisateur en fonction de la valeur ontologique des ressources en cour.

Enfin, pour démontrer la validité de nos propositions, nous définissons la plateforme I3 (Intelligent Interactive Intermediaries), une architecture qui supporte les différents mécanismes abordés tout en s'intégrant avec un minimum d'impact dans les

infrastructures existantes. Nous définissons le concept de proxlet qui permet une généralisation des agents intermédiaires pour la mise en place de services à l'utilisateur. Nous démontrons également l'intérêt de notre démarche en exposant les opportunités que notre plateforme apporte. Nous exploitons ainsi l'interaction rendue possible par la disponibilité du paradigme de session pour mettre en place des fonctionnalités ainsi que par une modélisation plus évoluée du cache: topologie, valeur sémantique, ontologie. Cette architecture permet l'implémentation dans les proxys de services tels que l'aide à la navigation, la mobilité, le filtrage et l'intégration de services.

ABSTRACT

Since the 90's, the Internet has tremendously evolved in terms of number and diversity of available services. In this trend, proxies are playing a central role and are spread all over the net. Today, the only functionality of proxies is to speed content access through caching only considering statistics based on use made by a user group, all together. Nowadays, evolutions are focusing on two main trends: specific infrastructures that do not comply anymore with the Web universality or individual applications like filtering. We propose to make proxies evolve from simple passive intermediaries to an open platform with advanced caching functionalities stressing on maintaining a high level of compatibility with existing paradigms. We ground our developments on two qualities already available in proxies, widely spread over the Net: the privileged position (plesiocentrism) within the network infrastructures and the perceptivity we extend to new dimensions, more specifically semantic (engnose).

In a first step, we elaborate a new Web resources indexation binding to the documents a semantic space, based on the absolute position defined from the URL and a relative position defined with the links that connect those resources. We extend the usage statistics to those new dimensions to build up a topological space that take into account the localization of the documents and the user browsing into that space. We present algorithms and functionalities to build, maintain and take advantage of this topology. With the aim of accelerating web browsing, we use this statistical space to implement a prefetching system based on Markov's model.

To extend the proxy perceptivity upstream to the users, we propose a mechanism to maintain a session, based on proxy-cookies. Therefore, we propose two new HTTP directives similar to those used for cookies. This paradigm also allows us to install personalized services with the support of the interaction concept and user profile. This last one allows us to tackle with mobility problems and to install proxies independently of physical network infrastructures.

In a second move, we study how the proxy could take into account new dimensions in the semantic web and ontology context. The new technologies emerging like XML and annotations bring new information. That information can easily be processed by a computer system. As for the annotations, they considerably enrich the available informations in the proxy's perception through the classification or resources in ontology. We underline the opportunity for proxys, regarding their situation, to integrate functionalities of annotation server. We demonstrate what can be acquired that way and advantages to be won with the engnose definition as a new proxy quality that become perceptive to web disseminated knowledge. We present a new cache management based on virtual multi-level cache. We present an algorithm able to switch automatically to the correct domain regarding the ontological value of currently visited resources.

Last, to demonstrate the validity of our propositions, we define the I3 platform (Intelligent Interactive Intermediaries): an architecture that supports all the various mechanisms presented above, but preserving integration with a minimum of impact on existing infrastructures. We define the concept of proxlet that is an intermediary agent generalization and allow the development of new user services. We demonstrate the value of our proposition by presenting opportunities brought by our platform. We take advantage of the interaction allowed by the session concept to implement functionalities also made possible by an advanced cache model based on topology, semantic and ontology. This platform allows the implementation of services such as browsing help, mobility filtering and service integration.

TABLE DES MATIÈRES

GLOSSAIRE	1
INTRODUCTION	5
1 Historique	5
2 Cadre de la thèse.....	6
3 Rôle des proxys	6
4 Objectifs	8
4.1 Démarche	10
4.1.1 Sémantique et ... sémantique	11
4.2 Applications	11
5 Structure du document.....	12
5.1 Introduction.....	13
5.2 Etat de l'art.....	13
5.3 Indexation sémantique	13
5.4 Sessions proxy.....	13
5.5 Web Sémantique.....	13
5.6 Proxy métier.....	14
5.7 La Plateforme I3	14
5.8 Conclusion	14
6 Mise en garde	14
ETAT DE L'ART	17
1 Introduction	17
2 Principes généraux.....	17
3 Le protocole HTTP	21
3.1 Historique	21
3.2 Introduction.....	21
3.3 Requêtes et réponses	22
3.4 Les directives.....	23
3.5 Caching	24
3.6 Rapport statistique	25
4 La Gestion du cache	25
4.1 Dimensions.....	27
4.2 Métriques pour la mesure des performances	28
4.2.1 Rendement.....	28
4.2.2 Coût/Bénéfice	29
5 Etat de l'art et classification des politiques de gestion	30
5.1 Politique simple d'ordre 1	30
5.1.1 Size.....	30
5.1.2 Autres paramètres	30
5.2 Politiques statistiques d'ordre 1	31
5.2.1 LRU - Last Recently Used	31
5.2.2 LFU - Least Frequently Used	31
5.2.3 Page load delay	31
5.2.4 EXP1	31
5.2.5 LRV - Lowest Relative Value (LRV)	31
5.3 Politique d'ordre n	31
5.3.1 Log-Size	31
5.3.2 Greedy-Dual-Size (with frequency): GD-Size et GDSF.....	31
5.3.3 Least Frequently Used with dynamic Aging (LFU-DA)	32
5.3.4 Hyper-G.....	32
5.3.5 LNC-R-W3-U	32
5.4 Politique à étages : caches virtuels.....	32
5.5 Hardware	32
5.6 Synthèse.....	33
5.6.1 Persistance des index et aberrations.....	33
6 Prefetching.....	34

6.1	<i>Chaînes de Markov</i>	34
6.2	<i>Volumes</i>	35
6.3	<i>Synthèse</i>	35
7	Statistiques d'utilisation	36
8	Conclusion	36
INDEXATION SEMANTIQUE		39
1	Introduction	39
1.1	<i>L'espace Web</i>	40
1.1.1	Le Webgraph	40
1.1.2	La position des documents sur la toile	41
1.2	<i>Indexation du cache en fonction de la localisation des objets</i>	42
1.2.1	Le Webgraph	42
1.2.2	Topologie basée sur les URL	43
1.2.3	Renforcement de l'activation de voisinage	45
1.3	<i>Applications à la gestion du cache</i>	45
2	Le prefetching	45
2.1	<i>Les chaînes de Markov</i>	46
2.2	<i>Adaptation au Webgraph</i>	47
3	Conclusion	48
SESSIONS PROXY		49
1	Introduction	49
1.1	<i>Définition</i>	49
2	HTTP et session	50
2.1	<i>L'authentification http</i>	51
2.2	<i>Les champs cachés</i>	52
2.3	<i>La réécriture des URL</i>	53
2.4	<i>Les cookies</i>	53
2.4.1	Fonctionnement des cookies	53
2.4.2	Limitations, problèmes	55
2.5	<i>Conclusion</i>	55
3	Proposition d'un système de maintien de session pour les proxys	56
3.1	<i>Les proxy-cookies</i>	56
3.1.1	Arborescence de proxys	57
3.1.2	Sécurité	58
3.1.3	Définition formelle	59
4	Conclusion	60
PROXY ET WEB SEMANTIQUE		61
1	Introduction	61
2	XML	62
2.1	<i>Langage de structuration des données</i>	62
2.2	<i>Sémantique des liens : XLink</i>	63
3	Annotations	64
3.1	<i>Annotations et méta-données</i>	65
3.2	<i>Localisation</i>	65
4	Composition	65
4.1	<i>Edge Side Includes</i>	66
4.2	<i>XML</i>	66
4.3	<i>Extension à d'autres médias</i>	67
4.3.1	Multimédia	67
4.3.2	SIG	67
5	Gestion de la connaissance: les ontologies	68
5.1	<i>Partage de la connaissance</i>	68
5.1.1	Communauté culturelle	68
5.1.2	Partage des conventions	69
5.1.3	Partage de la connaissance et des ontologies	69
5.2	<i>Annotations et ontologies</i>	69
5.3	<i>Proxy et localisation</i>	70
5.4	<i>Engnose</i>	71
6	Conclusion	71

PROXYS METIER.....	73
1 Introduction	73
2 Caches métier	73
2.1 <i>Nécessité de disposer de caches métier</i>	74
2.1.1 Distribution de Zipf	74
2.1.2 Cache métier	76
2.2 <i>Onto-plexing</i>	76
2.2.1 Isotopie sémantique étendue	77
2.2.2 Isotopie ontologique.....	77
2.3 <i>Application aux proxys</i>	78
2.3.1 Distance isotopique	78
2.3.2 Autodétermination du domaine actuel de l'utilisateur	79
2.4 <i>Conclusion</i>	80
3 Proxy métier: le cas SIRANAU	80
3.1 <i>Contexte</i>	80
3.2 <i>Méta-données</i>	81
3.3 <i>Distribution des sons</i>	81
3.4 <i>Utilisation des proxys pour SIRANAU</i>	81
4 Conclusion.....	82
LA PLATEFORME I3	83
1 Introduction	83
2 Architecture	84
2.1 <i>Opérations</i>	84
2.2 <i>Proxlet</i>	85
2.3 <i>Composants</i>	85
2.4 <i>Java</i>	86
3 Projets similaires.....	87
3.1 <i>Proxys adaptables</i>	87
3.1.1 JigSaw	87
3.1.2 MOWS	87
3.1.3 Muffin	88
3.1.4 IBM-WBI	89
3.1.5 Synthèse	90
3.2 <i>Outils d'annotation</i>	91
3.2.1 Annozilla	91
3.2.2 OntoMat-Annotizer	91
3.2.3 Synthèse	92
4 Modélisation du cache.....	92
4.1 <i>Ressource et propriétés</i>	93
4.2 <i>Ressource et environnement</i>	93
4.3 <i>Informations complémentaires</i>	95
4.4 <i>Conclusion</i>	95
5 Implémentation.....	95
5.1 <i>TomProxy</i>	96
5.1.1 Avantages, défauts	96
5.1.2 WBI.....	96
5.2 <i>Sessions</i>	96
5.3 <i>Affichage et interaction</i>	96
5.4 <i>Prototype</i>	97
6 Applications.....	97
6.1 <i>Filtre</i>	98
6.1.1 Protection parentale	98
6.1.2 Surlignage.....	98
6.1.3 Navigation anonyme	98
6.1.4 Filtrage de la publicité.....	98
6.2 <i>Aide à la navigation</i>	98
6.2.1 Graphe de navigation	99
6.2.2 Visualisation avancée des liens	100
6.3 <i>Business intelligence</i>	100
6.3.1 Document rating	100
6.3.2 Outil d'analyse type tableau de bord.....	100

6.4	<i>Mobilité</i>	101
6.4.1	Délégation des cookies.....	101
6.4.2	Délégation des bookmarks.....	101
6.4.3	Mobilité Intercache: Roaming Profile.....	101
6.4.4	Mobilité et collaboration du cache: Roaming Cache.....	102
6.5	<i>Intégration de services</i>	102
6.5.1	Annotations.....	102
6.5.2	Partage des bookmarks.....	102
6.5.3	Search focus.....	102
6.5.4	Migration.....	103
6.6	<i>Cache +</i>	103
7	Conclusion.....	103
SYNTHESE ET CONCLUSION.....		105
1	Résumé du contexte.....	105
2	Résumé de la démarche.....	106
3	Contributions.....	106
3.1	<i>Gestion du cache</i>	106
3.2	<i>Prefetching</i>	107
3.3	<i>Services intermédiaires</i>	107
3.4	<i>Proxlet</i>	107
4	Critiques.....	107
5	Prolongements.....	108
5.1	<i>Elargissement à d'autres rôles</i>	108
5.2	<i>Elargissement à d'autres protocoles</i>	109
5.2.1	Streaming.....	109
5.2.2	Telnet.....	109
6	Bilan.....	110
BIBLIOGRAPHIE.....		111

TABLE DES ILLUSTRATIONS

Figure 1 : Evolution du Web	7
Figure 2 : Le "Post-Scarcity World"	9
Figure 3 : Vue d'ensemble du plan.....	12
Figure 4 : Fonctionnement du proxy.....	17
Figure 5 : Situation générale.....	18
Figure 6 : Caching proxy.....	18
Figure 7 : Fonctionnement du caching.....	19
Figure 8 : Hiérarchie de proxys	20
Figure 9 : Format des requêtes	22
Figure 10 : Format des réponses	23
Figure 11 : Directives HTTP 1.0.....	24
Figure 12 : Directives HTTP 1.1 qui concernent les proxys.....	24
Figure 13 : Cache à étages.....	32
Figure 14 : WebGraph	43
Figure 15 : Chaîne de Markov, graphe et matrice de transition	46
Figure 16 : HTTP stateless	50
Figure 17 : Authentification HTTP	52
Figure 18 : URL avec paramètre	53
Figure 19 : les cookies: scénario	54
Figure 20 : Proxy-cookies: scénario	57
Figure 21 : Proxy-cookies: scénario avec hiérarchie de proxys	58
Figure 22 : Exemple d'utilisation des annotations	64
Figure 23 : Cache à étages et partitions métier	76
Figure 24 : Onto-plexing automatique	80
Figure 25 : Intervention sur les processus	84
Figure 26 : Processus de gestion de sessions	84
Figure 27 : Agent générateur.....	85
Figure 28 : I3, schéma des composants	86
Figure 29 : Chaîne de traitement des requêtes dans WBI.....	89
Figure 30 : Composants WBI.....	90
Figure 31 : Modèle "classique"	92
Figure 32 : Ressource et propriétés.....	93
Figure 33 : Ressource et environnement	94
Figure 34 : Annotations et domaine	95
Figure 35 : Séquence d'exécution à l'initialisation	97
Figure 36 : Exemple de représentation graphique possible pour l'aide à la navigation	99

GLOSSAIRE

CDN : Content Delivery Network (Réseau Fournisseur de Contenus). Un arrangement d'éléments Web destinés à maximiser l'efficacité de la distribution de contenus numériques.

Cookie : information qui est stockée par l'agent utilisateur à la demande d'un serveur. Cette information est ensuite, sous certaines conditions, retournée au serveur ou à un groupe de serveurs. Elle peut agir comme un marquage du client par le serveur.

Engnose : faculté de percevoir la connaissance.

FIFO : First In, First Out. Désigne tous les mécanismes de gestion de stockage (le plus souvent des queues), où le premier entré est le premier sorti.

HTTP : Hyper Text Transfer Protocol. Le protocole utilisé pour le transport des données dans le cadre du World Wide Web.

ICP : InterCache Protocol. Protocole permettant à des caching proxys de collaborer et d'échanger des ressources.

Interactif : qui supporte l'interactivité, sous-entend de supporter des interactions avec une personne physique dans le cadre d'une session utilisateur.

Internet : souvent défini comme le réseau des réseaux. Une infrastructure réseau mondiale qui regroupe différents protocoles et types de réseaux interconnectés. Supporte en général le protocole IP (Internet Protocole) et tous ses éléments ont une adresse IP qui permet de les identifier.

Intranet : réseau interne (à une entreprise) qui est connecté à Internet et qui se base sur des technologies Internet, mais dont les éléments ne sont pas ou partiellement accessibles depuis l'extérieur.

Extranet : l'ensemble des ressources d'un Intranet qui sont accessibles depuis l'extérieur de manière publique ou contrôlée.

ISP : Internet Service Provider. Fournisseur d'accès Internet.

LAN : Local Area Network. Réseau local.

LFU : Least Frequently Used. Désigne une politique de gestion de stockage qui élimine d'abord les objets les moins fréquemment utilisés.

LRU : Least Recently Used. Désigne une politique de gestion de stockage qui élimine d'abord les objets les moins récemment utilisés.

Plésiocentrique : situé au centre d'un voisinage, d'une communauté d'utilisateurs.

RFC : Request For Comment. Processus de normalisation mis en place dans le cadre du projet ARPANET (ancêtre d'Internet) qui définit donc les normes en vigueur pour Internet.

Session : suite d'interactions entre deux points de communication qui a lieu dans l'intervalle d'une seule connexion.

Session utilisateur : session faisant intervenir une personne physique pour l'un des points de l'interaction. La durée de vie de la connexion n'est plus liée à une connexion matérielle, mais à la perception de l'utilisateur.

SGBD : Système de Gestion de Bases de Données.

URI : Universal Resource Identifier. Classe abstraite pour désigner soit un URL, soit un URN

URL : Universal Resource Locator. Un pointeur sur une adresse unique du Web. Désigne de manière univoque une ressource Web.

URN : Universal Resource Name. Un identifiant unique pour une ressource, mais qui ne désigne pas comment accéder au document. Les URN ont été introduits pour pallier le problème de persistance des URL en cas de déplacement d'une ressource.

W3C : World Wide Web Consortium. Organisme en charge du développement et de la normalisation des technologies du Web.

Spécification HTTP (RFC 2616: [12])

Connexion : un circuit virtuel de la couche transport qui est établi entre deux programmes dans le but d'une communication.

Message : l'unité de base dans les communications HTTP qui consiste en une séquence structurée d'octets correspondant à la syntaxe définie et qui est transmise via une connexion.

Requête : un message HTTP qui constitue une requête selon les spécifications.

Réponse : un message HTTP qui constitue une réponse selon les spécifications.

Ressource : un objet ou un service numérique sur le réseau qui peut être identifié par un URI. Les ressources peuvent être disponibles dans plusieurs représentations (ex. langues, formats des données, taille et résolution) ou encore variables en d'autres façons.

Entité : l'information qui est physiquement transférée lors d'une requête ou d'une réponse. Une entité est constituée de méta-informations en tant que champs de l'en-tête de l'entité et le contenu en tant que corps de l'entité.

Représentation : une entité comprise dans une réponse qui a fait l'objet d'une négociation de contenu. Il peut exister plusieurs représentations associées avec un type particulier de réponse.

Négociation de contenu : le mécanisme de sélection pour une représentation appropriée lors de la réponse à une requête. La représentation d'une entité peut être négociée dans tout type de réponse (y compris pour une réponse signalant une erreur).

Variante : une ressource peut avoir une ou plusieurs représentations associées à tout moment. Chacune de ces représentations est appelée une variante.

Client : un programme qui établit une connexion dans le but d'envoyer des requêtes.

Agent utilisateur : le client qui est à l'origine des requêtes. Il s'agit souvent de navigateurs, d'éditeurs ou de tout outil à disposition d'une personne physique (utilisateur final).

Serveur : un programme qui accepte des connexions dans le but de répondre à des requêtes par l'envoi de réponses. Un quelconque programme peut être capable de jouer à la fois le rôle du client et du serveur. L'utilisation de ce terme indique uniquement le rôle joué par le programme pour une connexion spécifique et non une caractéristique générale du programme. Ainsi chaque serveur peut agir comme un serveur d'origine, un proxy, une passerelle, passant d'un comportement à l'autre en fonction de la nature de chaque requête.

Serveur d'origine : le serveur où réside la ressource en question ou sur le point d'être créée.

Proxy : un programme intermédiaire qui agit à la fois comme serveur et comme client dans le but de transférer des requêtes transmises par des clients. Les requêtes sont traitées de façon interne ou retransmises à d'autres serveurs avec d'éventuelles transformations. Un proxy doit à la fois implémenter les spécifications du client et celles du serveur. Un proxy transparent est un proxy qui ne modifie pas les requêtes ou les réponses au-delà de ce qui est nécessaire pour son authentification et son identification. Un proxy non transparent est un proxy qui modifie les requêtes ou les réponses dans le but de fournir des services supplémentaires à l'agent utilisateur comme par exemple des fonctionnalités d'annotation, des conversions du type de média, des réductions de protocole ou encore du filtrage pour rendre anonyme.

Passerelle : un serveur qui agit comme un intermédiaire pour d'autres serveurs. Contrairement à un proxy, une passerelle reçoit des requêtes comme si elle était elle-même le serveur d'origine. Le client qui adresse la requête peut très bien ne pas être conscient qu'il communique avec une passerelle.

Cache : un programme de stockage des messages réponses et le sous-système qui contrôle ce stockage des messages, leur récupération et leur suppression. Un cache stocke des

réponses cachables dans le but de réduire le temps de réponse et d'économiser la consommation en bande passante pour des requêtes futures équivalentes. Chaque client ou serveur peut inclure un cache.

Cachable : une réponse est cachable si le cache est autorisé à stocker une copie du message réponse pour l'utiliser comme réponse à des requêtes ultérieures. Les règles pour déterminer la cachabilité d'une réponse HTTP sont spécifiées. Même si une ressource est cachable, il peut y avoir d'autres contraintes limitant la possibilité pour un cache d'utiliser une copie cachée pour une requête particulière.

Première main : une réponse est dite de première main si elle provient directement et sans délai inutile du serveur d'origine, éventuellement via un ou plusieurs proxys. Une réponse est également de première main si sa validité vient d'être vérifiée directement avec le serveur d'origine.

Expiration explicite : le moment défini par le serveur d'origine au-delà duquel une entité ne devrait plus être retournée par un cache sans une nouvelle validation.

Expiration heuristique : une expiration définie par le cache quand il ne dispose d'aucune expiration explicite.

Age : l'âge d'une réponse est l'intervalle de temps écoulé depuis qu'elle a été envoyée ou validée par le serveur d'origine.

Durée de validité : l'intervalle de temps entre la génération d'une réponse et son expiration.

Valide : une réponse est valide si son âge n'excède pas la durée de validité.

Invalide : une réponse est invalide si son âge excède sa durée de validité.

Sémantiquement transparent : un cache a un comportement sémantiquement transparent, dans le cas d'une réponse en particulier, quand son action ne modifie ni le client qui émet la requête ni le serveur d'origine, hormis par l'accélération des performances. Quand un cache est sémantiquement transparent, le client reçoit exactement la même réponse (à l'exception des en-têtes signalant les points de passage) qu'il aurait reçue si la requête avait été traitée directement par le serveur d'origine.

Validator : un élément du protocole (ex: un tag sur une entité ou une indication de dernière modification) dont l'usage permet de déterminer si une entrée du cache est une copie équivalente à une entité.

Amont/Aval : décrivent le flux des messages: tous les messages circulent de l'amont vers l'aval.

Aller/Retour : détermine le chemin parcouru par les messages: "aller" signifie en direction du serveur d'origine alors que "retour" désigne le trajet en direction de l'agent utilisateur.

INTRODUCTION

Présentation du contexte et de l'objectif de ce travail

1 Historique

Depuis son invention en 1989 par Tim Berners-Lee [6], le World Wide Web s'est développé de manière exponentielle et dans deux directions : d'une part la prolifération des sites ainsi que des documents disponibles et accessibles au travers du protocole http, d'autre part (et dans un deuxième temps) la multiplication des services disponibles.

Face au risque d'engorgement d'Internet, des solutions auparavant esquissées pour d'autres types de services en réseau comme FTP se sont très rapidement imposées et généralisées. Il s'agit des caching proxys, que l'on désigne parfois abusivement par le seul terme de proxys.

Dès les premières propositions dans les années 1990 pour gérer les documents scientifiques du CERN, l'idée de cache fait son apparition [13] dans une optique de gain en performances, mais l'auteur n'a alors à l'esprit que le cache local à l'application client. Ainsi MOSAIC, le premier¹ navigateur WWW [14], propose en 1993 une fonctionnalité de caching en copiant les objets auxquels on a accédé dans un espace disque utilisateur, au fur et à mesure de la navigation. Si, par la suite, l'utilisateur accède à nouveau aux mêmes objets, le navigateur (sous certaines conditions que nous détaillerons plus loin) recharge alors les objets du cache pour les afficher.

En 1993, Peter Danzig propose une structure de cache serveur pour augmenter les performances et réduire le trafic FTP, dont l'importance en terme de trafic est encore nettement supérieure au protocole HTTP. Le principe décrit dans [15] est de mettre en place un serveur intermédiaire, appelé proxy, auquel sont adressées toutes les requêtes. Le proxy se comporte également comme un client en relayant les requêtes vers les serveurs d'origine et en copiant les objets auxquels on a accédé au fur et à mesure dans un espace disque : son cache. Si ensuite l'un d'eux est à nouveau requis, il prend alors (de nouveau sous certaines conditions) l'objet caché. Cette architecture permet à plusieurs utilisateurs de partager un même cache géré par le proxy.

Le mot "proxy" est un terme anglais qui vient du domaine juridique et désigne un mandataire, un fondé de pouvoir. En informatique, ce terme désigne tout ce qui constitue un intermédiaire, au niveau applicatif entre deux agents. On trouve ainsi bien évidemment des proxys dans les applications réseaux mais également en programmation objet où un proxy représente une interface d'accès à un objet. Dans la programmation objet, un proxy permet d'accéder à un agent sans avoir à se soucier de détails physiques ou de la localisation de l'objet dans des environnements distribués.

En avril 1994, Luotonen [16] propose une première description des WWW Proxys dont l'objectif initial est de permettre de « passer » les firewalls configurés pour interdire l'accès des clients vers l'extérieur. L'idée de cache est déjà présente : « A brand new feature is caching performed by the proxy ».

¹ Mosaic n'est pas à proprement parler le premier navigateur Web, puisque des prototypes ont été développés avant au CERN pour valider le concept de WWW. Mais c'est le premier navigateur suffisamment élaboré avec une interface graphique qui a séduit le public et démarré l'expansion d'Internet.

Le premier système de cache Harvest apparaît en 1995 dans le domaine open-source. Harvest n'implémente pas seulement une fonction de caching (fonction qui n'est d'ailleurs pas dans les buts initiaux du projet) mais un système d'indexation des documents Web qui propose une interface d'échange d'index avec d'autres systèmes similaires et des outils de recherche pour l'utilisateur (voir [17] et plus loin pour plus de détails). Le projet Harvest est abandonné au profit de Squid dès 1996. Il est aujourd'hui le proxy le plus répandu et ne reprend d'Harvest que la partie caching. Depuis 1996, le premier serveur http, le CERN HTTPd, propose également la fonctionnalité de caching proxy. Mais, moins performant que Squid, il ne sera que peu utilisé comme tel pour des cas où l'on souhaite disposer d'une seule application qui fasse à la fois serveur Web et proxy (cas des petits ISP).

Par la suite apparaissent des produits commerciaux, dont certains se spécialisent ou apportent des changements d'architecture. Entre autres, Microsoft et Cisco intègrent des caching proxys dans leur suite de produits. Inktomi s'oriente vers les solutions intranet et l'indexation alors qu' Akamai s'oriente vers des solutions s'apparentant au server-push qui aboutissent au concept de Content Delivery Network (CDN) [18]. Akamai combine la transparence des proxys avec la distribution proactive des objets dans le voisinage des utilisateurs à l'instar des serveurs miroirs.

En 1997, IBM présente le concept de WBI (Web Browser Intelligence) [19, 20], qui propose une plateforme ouverte pour l'installation de nouveaux services à l'intention d'un (seul) utilisateur. Ce sont les premiers travaux qui ne se focalisent pas sur une problématique particulière, mais s'appuient sur les proxys pour proposer une plateforme ouverte.

C'est le 1^{er} Octobre 1996 que se tient à Varsovie la première conférence « Web Caching Workshop ». En 2000, la conférence s'intitule : « Web Caching and Content Delivery Workshop » et s'ajoute alors le concept de (CDN).

2 Cadre de la thèse

Le terme "proxy" est un terme générique, comme le montre la définition que nous avons donnée. Il désigne toute forme d'intermédiaire entre deux agents. Dans la suite de notre travail, nous ne nous intéresserons qu'aux proxys Internet, c'est-à-dire aux intermédiaires dans des opérations client serveur, ou plus généralement dans un contexte réseau. Ce domaine est riche en protocoles et en services, nous avons concentré notre travail sur les problématiques rattachées au développement du Web. Nous n'avons pas exclu les autres champs d'application des proxys et ils existent, à commencer par le protocole FTP qui a été historiquement le premier domaine où les proxys ont commencé à se développer. Mais nous n'avons pas pris en compte ces domaines dans nos recherches parce que le potentiel de développement n'était pas aussi riche. Cependant, les résultats que nous avons obtenus dans ce cadre restreint peuvent être étendus, nous en aborderons quelques possibilités en conclusion.

Dans la suite du travail, le terme "proxy" désigne donc uniquement un agent intermédiaire dans des transactions client serveur et, sauf indication contraire, dans des transactions supportées par le protocole HTTP.

3 Rôle des proxys

Le rôle des proxys est de permettre d'une part d'économiser de la consommation en bande passante puisque, dans le cas idéal, pour plusieurs requêtes d'un même document, une seule aboutit à une requête qui "sort" du réseau local. D'autre part, pour l'utilisateur, le proxy permet une accélération dans la mise à disposition des documents qui se

trouvent dans le cache. On induit ainsi une augmentation de la vitesse apparente de téléchargement des documents.

Cependant, le caractère exponentiel de la prolifération des documents disponibles sur le Web a rapidement motivé les principaux acteurs d'Internet à trouver des solutions pour éviter la saturation et le ralentissement du réseau. En effet, le protocole http s'est très largement imposé comme standard de transport de données pour la plupart des applications client/serveur dès le milieu des années 90. La simplicité du développement et du déploiement en fait un choix quasi évident dans la plupart des cas. De plus, l'apparition de nouveaux outils comme le langage Java et la multiplication des bibliothèques favorisent encore le choix de baser les nouvelles applications sur les technologies du Web, que ce soit dans le contexte bancaire, industriel, etc. Mais la prolifération des services échappe à ce jour encore aux formes d'optimisation qui pourraient être installées actuellement au niveau des infrastructures du réseau.

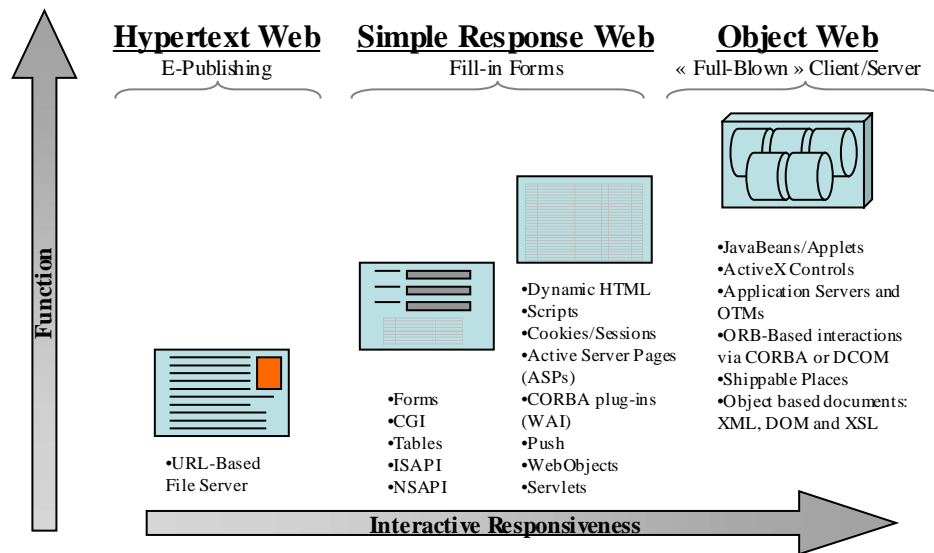


Figure 1 : Evolution du Web

Les caching proxys ont très rapidement répondu à certains problèmes et ont été largement déployés sur Internet. Cette première forme, très pratique, répond à un certain nombre d'exigences: diminution de la consommation de la bande passante et accélération de la vitesse apparente du réseau. Mais la situation est loin d'être idéale et l'on répertorie de nombreux problèmes ou limitations associés aux proxys (voir [21, 22]). Ces problèmes sont liés à l'évolution des documents distribués sur le Web (audio, vidéo, etc.) et surtout à l'augmentation de l'interactivité entraînée par une mise en Web de services de plus en plus évolués. Ainsi, au fur et à mesure que le Web évolue (Figure 1, tirée de [23]) de la simple distribution de documents vers la mise en ligne d'agents, l'efficacité des caching proxys se détériore.

Face à cette évolution, on constate grossièrement deux types d'approches. D'un côté, des normes qui s'inscrivent dans un ensemble proposé par le W3C avec par exemple toute la panoplie des spécifications concernant XML, le passage de HTTP 1.0 à HTTP 1.1 et, de l'autre, des solutions concurrentes qui divergent des standards ouverts d'Internet et proposent des réponses adaptées à des besoins particuliers, mais restreignant

drastiquement l'universalité du Web. Dans cette dernière catégorie, on trouve par exemple le cas du langage ESI (Edge Side Include que nous décrivons plus loin). Ce mouvement traduit le besoin de mettre en place des solutions plus locales pour des applications spécifiques. Ceci dans un mouvement d'exploitation des intranets, où des solutions particulières et locales s'appuient sur les technologies de l'Internet.

Dans ce contexte, les proxys ont assurément un rôle à jouer. Ils sont très largement répandus puisqu'on en trouve quasiment à toutes les portes de sortie des intranets. Mais pour le moment, la principale limitation des proxys réside dans un manque d'adaptation et d'ouverture vers leur voisinage le plus direct : la communauté dont ils sont le centre. Si nous pouvons lever ces difficultés, alors nous enrichissons une plateforme qui existe en de nombreux points du réseau. Avec ces enrichissements apportés à un service largement répandu, nous serons en mesure de proposer des améliorations et de nouveaux services à de très nombreuses communautés, si ce n'est à l'ensemble des internautes.

Il existe bien sûr d'autres points de passage disséminés dans les architectures réseaux: router, firewall, tunnel, etc. Ceux-ci se situent cependant plus bas dans les couches OSI (transport, réseau) alors que les proxys travaillent au niveau du protocole HTTP et donc de la couche application.

On pourrait se demander pourquoi nous avons choisi de placer exclusivement les proxys au centre de notre travail, et s'opposer à ce choix par l'argument suivant : ce que l'on attend avant tout d'un proxy c'est une amélioration des performances et une économie de trafic sur l'extérieur d'un LAN. Il existe d'autres solutions pour intégrer des services réseaux pour une communauté d'utilisateurs. Nous écartons tout d'abord les firewalls qui, s'ils disposent également d'une situation privilégiée sur les intranets, ne sont pas soumis à une normalisation aussi précise que les proxys (implémentation http), et ne supporteraient donc pas de généralisation.

Alors, pourquoi ne pas proposer l'élaboration d'un nouveau service indépendant? C'est l'optique prise pour les CDN. Elle présente certes l'avantage de pouvoir établir un système profilé en fonction des besoins sans avoir à s'encombrer de contraintes parfois peu pratiques. Cependant, elle a l'inconvénient de devoir déployer de nouveaux standards, de nouveaux protocoles et surtout de nouvelles applications client, alors que les caching proxys sont déjà utilisés par des milliers d'internautes sans qu'ils en aient toujours conscience.

4 Objectifs

Nous établissons le constat que les proxys occupent une position tout à fait privilégiée au sein des infrastructures de l'Internet puisqu'ils constituent un passage quasi obligé dans toutes les transactions qui prennent place entre une communauté d'utilisateurs et le "monde extérieur", que ce soit pour accéder à des documents ou des services. En reprenant l'argument " Whereas web servers have traditionally been responsible for producing all content, intermediaries now provide new places for producing and manipulating web data"² [20] nous établissons un parallèle avec les systèmes informatiques qui ont connu un premier développement centré sur les serveurs (mainframes) suivi de l'évolution des clients (PC) et enfin une dernière ère concentrée sur les middlewares et la généralisation des architectures N-Tiers. En examinant la Figure 2 tirée de [23], qui montre en quoi consiste la généralisation de l'usage des intermédiaires dans des architectures client/serveur on peut facilement substituer Middleware par Proxy. Dans notre contexte, les proxys constituent donc l'élément intermédiaire privilégié

² Partout où les serveurs Web ont traditionnellement été responsables de la production de contenus, les intermédiaires fournissent maintenant un nouvel emplacement pour produire et manipuler des données du web.

pour une même évolution avec une omniprésence qui tend à se confirmer sur le Web et une amorce de généralisation qui s'esquisse dans des solutions comme WBI (solution que nous détaillons au chapitre 7).

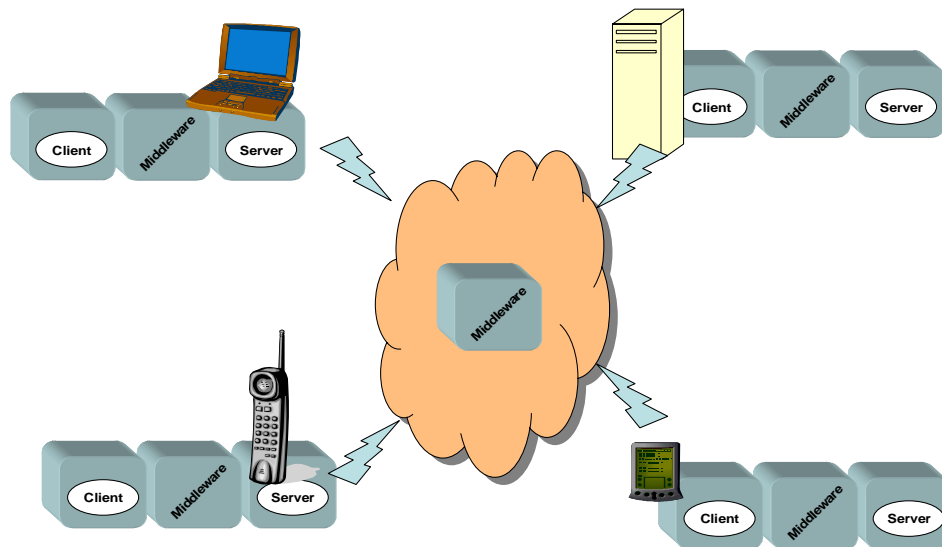


Figure 2 : Le "Post-Scarcity World"

Dans notre travail, nous établirons deux qualités principales des proxys relatives à leur situation et à leur perceptivité par rapport aux documents qui les traversent. Pour prendre une image de la vie courante, le proxy est comme le facteur: un élément central qui se définit par rapport aux quartiers qu'il sert et un acteur discret qui a une grande connaissance des habitudes globales et individuelles des habitants de ce même quartier. Nous définirons à cet effet le plésiocentrisme et l'engnose. Plésiocentrisme vient du grec $\pi\lambda\eta\sigma\acute{\iota}\omicron\varsigma$ (voisin) et signifie donc central au voisinage. Nous parlerons de perceptivité lorsque le proxy aura accès aux dimensions physiques du document et aux données statistiques d'utilisation. Mais lorsque cette perceptivité s'étend aux schémas de connaissance structurée qui émergent dans la structure du Web, nous définissons alors l'engnose. A l'instar de l'empathie, l'engnose définit la capacité cognitive par l'acquisition de la dimension ontologique des données véhiculées.

Nous démontrerons qu'en renforçant ces deux qualités dans les proxys, nous obtenons une plateforme qui permet une amélioration du caching par des politiques de gestion plus efficaces et la prise en compte de la valeur ontologique des documents.

De plus, nous donnons aux proxys la possibilité d'interagir avec les utilisateurs, ce qui leur adjoint un potentiel de services à la communauté et aux utilisateurs. L'interaction avec l'utilisateur présuppose le moyen d'établir une session avec celui-ci, concept qui n'est que partiellement supporté dans le protocole HTTP et surtout d'une façon non généralisable aux transactions établies avec un proxy. Nous proposons donc de pallier ce manque par la définition d'un mécanisme de maintien de session entre l'agent utilisateur et le proxy. Cette session permet l'interaction proxy/utilisateur et donc la définition de profils personnalisés. Ces derniers peuvent servir à supporter la mobilité, mais aussi à

collecter des informations détaillées, de la même manière que les organismes de poste utilisent la connaissance des facteurs à des fins de recensements commerciaux³.

4.1 *Démarche*

La première qualité des proxys, le plésiocentrisme, est une qualité inhérente qui tient dans la situation du service. C'est plus un constat qu'une qualité qui doit être développée. Cependant, si nous mettons l'accent sur ce point, c'est que cette qualité n'est que peu exploitée. Il se trouve que les proxys en bénéficient de facto, mais nous verrons que cette qualité amène au proxy un potentiel qui n'est pas développé à l'heure actuelle. C'est pour justifier l'exploitation de cette qualité inhérente que nous introduisons de nouveaux services dans le proxy. De plus, celle-ci appuie et renforce le développement des autres qualités.

La perceptivité des proxys n'a été utilisée, jusqu'ici, que pour le caching. D'abord limitée à l'identifiant des documents et aux statistiques d'accès à ceux-ci d'une communauté d'utilisateurs pour des polices de remplacement de type LFU (Least Frequently Used) ou LRU (Last Recently Used), elle s'est quelque peu étendue à d'autres propriétés des documents : type, taille, etc. Nous appellerons les différentes caractéristiques que perçoit le proxy des dimensions de son espace perceptif. Celles-ci sont soit inhérentes aux documents, soit induites par l'utilisation de la communauté d'utilisateurs du proxy, ce qui nous ramène au plésiocentrisme.

Afin d'affiner la gestion du cache, nous tenterons d'étendre les dimensions accessibles en nous basant notamment sur la localisation du document induite par la sémantique de l'URL et la position relative des documents induite par les liens. De plus, cet espace de perception constitue une information qui ne fait pas partie explicitement des documents, mais qui peut cependant s'avérer pertinente. Ces informations ne sont à l'heure actuelle exploitées que pour la gestion du cache, d'où notre intention de les rendre exploitables également par les utilisateurs grâce à l'adjonction de services proxy.

Une des premières fonctions des proxys est d'accélérer la vitesse apparente du réseau. Dans cette optique, l'acquisition de nouvelles dimensions ne peut pas se faire au détriment de la vitesse de transfert des requêtes et des documents au travers du proxy. Nous n'envisagerons donc pas de processus de traitement lourds comme l'analyse sémantique du contenu des documents. Par contre, le Web sémantique nous apporte du sens dans des formes qui sont beaucoup plus "apparentes". En exploitant ces informations, nous pouvons ainsi étendre la perception du proxy à de nouvelles dimensions qui se rapportent à la gestion de la connaissance dans l'espace Web, notamment grâce aux ontologies. Nous appelons cette nouvelle perception l'engnose, qui signifie l'appropriation passive des connaissances comme nous l'avons définie auparavant.

Enfin, comme nous l'avons déjà signalé, l'évolution fulgurante du Web génère énormément de besoins contradictoires. Les proxys, dans leur forme classique, ont de la peine à s'adapter à cette évolution et l'on est obligé de constater que toujours plus de problèmes apparaissent et que des solutions divergentes sont proposées. Dans ce contexte, afin de pouvoir répondre rapidement aux inadéquations entre le proxy et de nouveaux concepts tout en gardant une solution unifiée, nous souhaitons ajouter de la flexibilité dans la conception des proxys.

En résumé, en s'appuyant sur le plésiocentrisme, notre travail vise à améliorer dans un premier temps la perceptivité des proxys puis à l'étendre à l'engnose, le tout pour pouvoir

³ En France en tout cas, on fait état dans certains médias d'un débat portant sur le fait que les facteurs opèrent des recensements discrets sur le nombre d'habitants, les habitudes, le niveau de vie, etc. des habitants desservis dans le but de campagnes publicitaires ciblées.

améliorer la fonction primaire du proxy qu'est le caching et pour proposer de nouveaux services à la communauté (voisinage) du proxy en donnant ainsi une certaine flexibilité à ces derniers.

De plus, ayant écarté les démarches qui ont conduit des développements comme les CDN avec une altération des standards, nous avons donc choisi de conserver la transparence d'utilisation des proxys, de minimiser au maximum les changements de protocole dont nous pourrions avoir besoin pour proposer de nouveaux services.

4.1.1 *Sémantique et ... sémantique*

Nous emploierons dans notre travail le terme "sémantique" pour désigner deux propriétés bien distinctes et il est important de ne pas confondre ces deux utilisations. Dans la première partie, ce terme est utilisé dans son acception usuelle: qui se rapporte au sens, à la signification. Dans un deuxième temps, dès que nous aborderons le Web sémantique, ce terme prendra alors la signification qui lui a été donnée dans ce contexte : qui peut être traité par une machine.

Si nous avons conservé cette ambiguïté, c'est parce que dans le premier cas, il n'est malheureusement pas possible de trouver d'équivalent juste et dans la deuxième acception, c'est la définition qui a été donnée par Tim Berners-Lee dans le cadre du Web sémantique, et donc celle qui fait autorité dans ce contexte.

Afin de ne pas perdre le lecteur, nous avons fait en sorte que le contexte permette de comprendre clairement de quel sens il s'agit. De plus, dans l'ensemble, le sens "classique" est le seul à prendre en compte dans les quatre premiers chapitres. Dès le moment où l'on introduit le Web sémantique au chapitre 5, on se trouve alors, sauf indication contraire, dans une utilisation correspondant à la définition de Tim Berners-Lee.

4.2 *Applications*

La première fonctionnalité des proxys est de servir d'intermédiaire pour amener une augmentation des performances. Cependant, leur rôle d'intermédiaire transparent constitue un potentiel de déploiement très varié, que ce soit pour du caching local, du filtrage ou encore de l'intégration de services. Dès le début de notre travail, nous avons envisagé un certain nombre de nouveaux domaines d'application où nous pourrions introduire des proxys.

Comme caching local, pour augmenter l'efficacité de la distribution de documents, nous trouvons un exemple dans un prototype mis au point pour l'archivage numérique audio d'une radio décrit dans [24]. Dans le cadre de ce projet, les temps d'accès posaient un problème critique en regard de la taille des documents. La détermination d'une utilisation par contexte délimité physiquement et donc la définition de voisinage permet immédiatement l'utilisation de proxys pour une augmentation apparente de la vitesse d'accès.

Nous n'avons parlé jusqu'ici que de l'utilisation du caching proxy du côté des agents utilisateurs. Il existe une utilisation symétrique, on parle alors d'accélérateur HTTP. Dans ce cas, le proxy se trouve alors installé dans le voisinage du serveur d'origine et il sert d'intermédiaire pour un seul serveur d'origine (ou éventuellement un groupe de serveurs déterminés dans des cas de clustering de serveurs Web). Dans cette configuration, le proxy a pour but d'accélérer l'accès à un service Web en se basant sur le principe que la gestion du stockage est plus rapide dans un proxy et qu'il exploite également mieux le caching d'objets en mémoire centrale. Cette architecture peut être étendue à des clusters de serveurs Web avec une gestion beaucoup plus compliquée au niveau du caching, si l'on souhaite étendre le caching à du contenu dynamique, pour cela se référer à [25, 26].

L'utilisation des proxys pour des opérations de filtrage a déjà été explorée. Des projets comme Muffin [27] en sont un bon exemple. Contrairement aux firewalls par exemple, le

fait de se situer au niveau de la couche application permet d'intervenir en fonction des caractéristiques des documents échangés.

La dissémination est un élément constitutif du Web. Si cette caractéristique est une richesse, elle se révèle aussi une faiblesse par la dilution de l'information. Le résultat est une désorientation pour l'utilisateur et un accès alambiqué dans ce qui constitue une jungle numérique. Dans ce contexte, la mise en perspective et la compilation de services distribués apportent une valeur ajoutée à l'information. Simplement en jouant le rôle d'intermédiaire, le proxy peut apporter des fonctionnalités d'intégration de services disséminés.

Signalons enfin, pour souligner que les possibilités des proxys sont loin d'être épuisées, une situation originale pour l'exploitation d'un caching proxy avec pour contexte les communications avec un satellite. On trouve ainsi dans [28] une proposition pour l'intégration d'un caching proxy Internet dans des communications transitant par un satellite INTELSAT.

Nous montrerons plus loin dans ce travail comment le proxy peut répondre à ces différents besoins, d'abord en proposant un certain nombre de mécanismes permettant d'implémenter ces différentes fonctionnalités, ensuite en proposant une plateforme et des exemples d'applications qui les illustrent plus en détail.

5 Structure du document

L'exploitation de deux qualités: plésiocentrisme et perceptivité puis engnose constituent donc le fil conducteur, et ceci dans le double objectif de l'amélioration du cache et de la mise en place de services. Notre document se construit par conséquent au travers de ces quatre éléments.

On retrouve les deux qualités comme fil conducteur au travers des différents chapitres, et la réalisation de notre double objectif dans la présentation de la plateforme I3 au chapitre 7. Cette structure est illustrée dans le schéma (Figure 3).

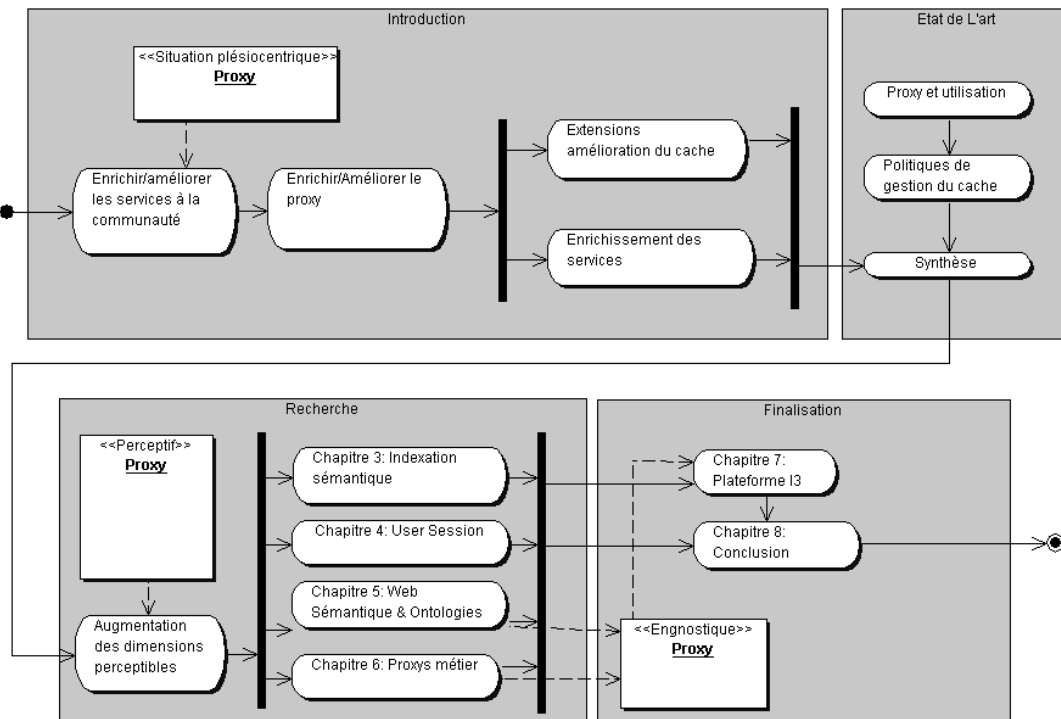


Figure 3 : Vue d'ensemble du plan

5.1 Introduction

Dans ce chapitre, nous présentons le domaine concerné par notre travail et les principes généraux dont il procède. Nous donnons également une première vue d'ensemble des enjeux qui ont motivé notre recherche en démontrant la place stratégique qu'occupent les caching proxys. Nous présentons enfin les lignes directrices que nous avons suivies pour notre travail en survolant les objectifs que nous nous sommes fixés.

5.2 Etat de l'art

Nous examinons en détail le fonctionnement des proxys et établissons les concepts et les définitions de base qui seront utilisés dans la suite du travail. Par une présentation chronologique des évolutions et des recherches faites dans le domaine, nous établissons également l'état de l'art en la matière. Enfin, nous exposons les limitations et les possibilités qu'offrent les proxys à ce jour, ce qui alimentera le reste de notre travail.

Nous présentons ici tout ce qui concerne la gestion du cache. En énumérant les différents paramètres qui entrent en jeu, nous présentons chaque fois les algorithmes associés qui ont été proposés, ainsi que les différentes métriques permettant d'établir les performances. Nous examinons également les études proposées sur une technique complémentaire au caching : le prefetching.

Les statistiques d'utilisation des proxys montrent un profil qui correspond à la loi de Zipf que nous présenterons car elle permet de modéliser le comportement des utilisateurs.

5.3 Indexation sémantique

Nous proposons une méthode d'indexation originale qui se base sur la construction d'un espace topologique sémantique. Pour cela, nous définissons de nouvelles dimensions qui se basent sur la position absolue extraite de la sémantique des URL des ressources du Web ainsi que la position relative induite par les liens entre les documents. Nous associons à cet espace les statistiques d'utilisation, et plus particulièrement les statistiques de cheminement des utilisateurs pour construire une topologie induite par le comportement des clients. Nous présentons les avantages de celle-ci ainsi que les bénéfices qui peuvent en être tirés aussi bien pour la gestion du cache que pour le prefetching.

5.4 Sessions proxy

Afin de permettre aux proxys de collecter des statistiques d'utilisation d'une granularité plus fine par rapport au comportement des utilisateurs en amont des transactions, nous définissons un mécanisme de maintien de session basé sur le même modèle que les cookies : les proxy-cookies. Celui-ci permet également d'établir des profils utilisateurs dans un objectif de mobilité et une interactivité permettant la mise en place de services au niveau du proxy.

5.5 Web Sémantique

Le Web sémantique englobe plusieurs domaines et techniques visant à améliorer l'utilisation du Web. De par sa définition, on démontre aisément le parti que peuvent en tirer les proxys et nous démontrons ici comment ils peuvent en bénéficier par leur position clé dans les infrastructures réseau et quels sont les éléments, en particulier dans les technologies du Web sémantique, dont ils pourraient tirer profit. L'objectif de ce chapitre est d'établir un rôle clé pour les proxys dans l'utilisation de techniques comme XML et les annotations. Nous définirons également le type d'enrichissement dont

peuvent tirer parti les proxys et l'exploitation qui peut en être faite pour une meilleure gestion du cache et la mise à disposition d'informations et de services avancés pour les utilisateurs.

5.6 *Proxy métier*

Nous développons deux axes dans ce chapitre. D'abord nous proposons l'exploitation des informations étendues par la mise en place d'une bascule de la gestion du cache dans le domaine courant de l'utilisateur en fonction des documents qui s'enchaînent dans la session en cours. En fonction du domaine courant, nous pouvons alors exploiter une structure de cache virtuel à étages qui nous permet de proposer des fonctionnalités de caching compartimenté par métier.

Dans la deuxième partie, nous verrons comment, dans un cas particulier, l'exploitation de proxys permet la mise en place intégrée aux services d'une accélération de ces derniers, sur des infrastructures réseau prédéfinies.

5.7 *La Plateforme I3*

Afin de valider les différents concepts que nous avons évoqués auparavant, nous présentons ici l'architecture d'une plateforme qui permet la mise en place de ce qui a précédé tout en respectant notre objectif initial, à savoir minimiser l'impact sur les paradigmes existants. Pour cela, nous définissons le proxlet comme généralisation d'agents intermédiaires et nous proposons ensuite six catégories d'applications que nous pouvons mettre en place dans notre plateforme pour faire évoluer les proxys : le filtrage, l'aide à la navigation, la business intelligence, la mobilité, l'intégration de services et enfin la mise en place de caching avancé.

5.8 *Conclusion*

Les domaines que nous avons abordés dans ce travail regroupent de très nombreuses technologies en constante évolution. Par conséquent, les possibilités sont encore infinies et nous proposons dans cette dernière partie de conclure en évoquant celles-ci par quelques exemples et en soulignant l'adéquation entre les concepts présentés dans ce travail et de futurs développements.

6 **Mise en garde**

Internet et plus particulièrement ce qui touche au World Wide Web fait l'objet de très nombreux efforts de recherche et de développement auxquels se consacre une communauté particulièrement large et variée. Il existe bien entendu des organismes de normalisation comme le W3C qui participent d'ailleurs activement à ces développements, mais on découvre presque quotidiennement de nouvelles propositions aux durées de vie incertaines. Dans ce contexte, nous considérons comme universellement définis et utilisés (entre autres !) : les protocoles HTTP, le langage HTML, le langage Java (normalisé par SUN) et les principes fondamentaux de XML. A noter qu'à propos d'XML, de nombreuses normalisations sont encore à l'étude, mais nous considérons comme stables les normes et les définitions établies par le W3C. Ceci ne constitue par contre en aucun cas un jugement de valeur sur les nombreuses propositions faites autour des technologies associées au Web.

Dans ce contexte effervescent, les proxys ne font pas directement en eux-mêmes l'objet de changements aussi rapides. Cependant, ils sont, de par leur nature, au centre des

nombreuses évolutions qui touchent le Web et les opportunités ainsi ouvertes sont évidemment aussi multiples.

Dans la suite de notre travail, nous avons fait un certain nombre de choix, notamment celui de nous concentrer sur les technologies évoquées ci-dessus. Ceux-ci sont bien évidemment contestables, mais dans l'ensemble, nous nous sommes concentrés sur l'exploitation des proxys sur la base de technologies majoritairement utilisées. Par exemple, nous avons choisi le protocole HTTP, alors que d'autres protocoles de transport comme FTP, Jabber [29], etc. sont certainement tout aussi intéressants et présentent de nombreuses opportunités pour le paradigme du proxy.

De même pour les outils, nous avons fait des choix en nous basant sur des critères parfois plus quantitatifs que qualitatifs (universalité) et sur la disponibilité pour de nouveaux développements. Ainsi, le modèle d'Akamai repose sur une infrastructure très large puisque des installations ont été disposées au niveau mondial pour la distribution de contenus Internet. Cependant, cette dernière technologie est propriétaire et appartient à une société commerciale.

Enfin, nous nous excusons auprès du lecteur qui ne trouvera pas dans ce travail telle ou telle technologie. Il en existe pléthore et nous n'aurions pas pu toutes les couvrir, surtout avec la vitesse de production que présente la communauté Internet.

ETAT DE L'ART

Les différentes technologies et développements associés aux proxys.

1 Introduction

Pour présenter l'état actuel des technologies et outils à disposition dans le domaine des caching proxys, nous examinons tout d'abord le fonctionnement général des proxys. Comme nous avons choisi de nous concentrer sur l'utilisation des proxys dans le cadre du protocole HTTP, nous donnons également une introduction générale sur cette technologie.

Ensuite, nous abordons les solutions existantes dans le domaine que nous avons sélectionné, qu'il s'agisse de solutions commerciales ou "open-source". Nous ne donnons pas une liste exhaustive, mais un éventail de solutions parmi les plus répandues.

Enfin, une partie importante concernant les proxys est le caching, puisque dans le contexte des applications Web, c'est la constituante principale qui a motivé les principaux développements dans ce domaine. La gestion du cache a fait l'objet de nombreuses recherches et nous en donnons ici l'essentiel.

Nous terminons avec une synthèse où nous présentons les faiblesses, les problèmes qui sont liés aux solutions actuelles disponibles.

2 Principes généraux

Le fonctionnement de base d'un caching proxy « classique », comme décrit dans [30] est relativement simple. Le client adresse toutes ses requêtes au proxy exactement de la même façon qu'il les adresserait au serveur d'origine (Figure 4).

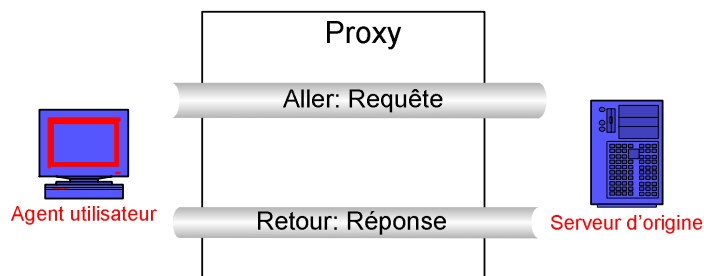


Figure 4 : Fonctionnement du proxy

Du côté du client, le proxy agit comme un serveur (retour d'une réponse à une requête reçue), alors que du côté serveur il agit comme un client (envoi d'une requête et attente d'une réponse en retour).

Du fait de ce comportement caméléon, le client n'a pas à changer son fonctionnement pour s'adresser à un proxy. Les requêtes d'un client peuvent alors être redirigées sur un proxy à son insu. On obtient généralement ceci en combinant un routeur et un proxy. Le routeur redirige automatiquement toutes les requêtes adressées sur le port 80 (qui est le

port normalisé pour les connexions HTTP) vers un proxy. Cette solution, appelée dans [30] "caching Web transparent", économise la configuration d'un proxy au niveau de l'agent utilisateur et permet de forcer l'utilisation du proxy. Par contre, toutes les requêtes HTTP qui sont faites sur d'autres ports (ce qui est parfaitement possible !) échappent alors au mécanisme.

En général, les proxys sont situés sur un réseau local (intranet), et sont partagés par une communauté d'utilisateurs. Ils constituent alors un point de passage pour les requêtes des différents agents utilisateurs qui se trouvent dans le réseau local (Figure 5).

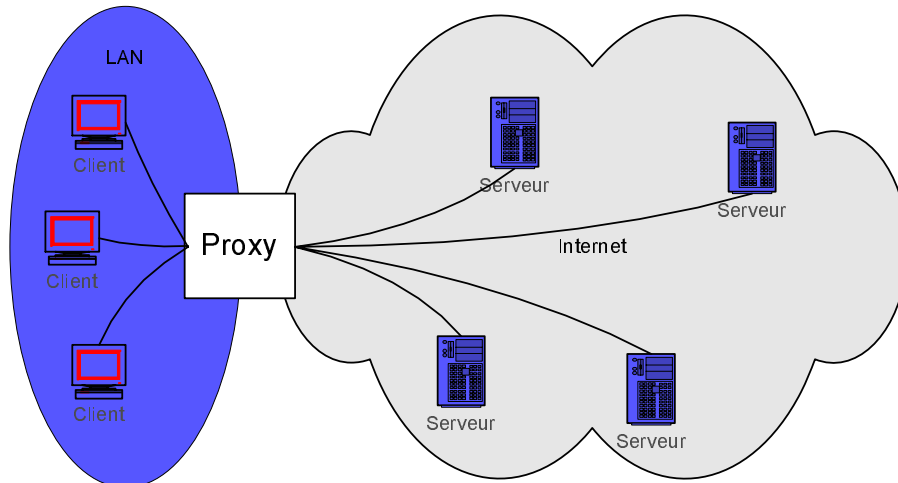


Figure 5 : Situation générale

Cette situation de point de passage au centre d'une communauté constitue la première qualité que nous avons définie précédemment : le plésiocentrisme⁴.

La fonctionnalité première des proxys est le caching. Il implique l'utilisation d'un espace de stockage où des copies des réponses sont déposées et, le cas échéant, servies aux utilisateurs lors de requêtes ultérieures.

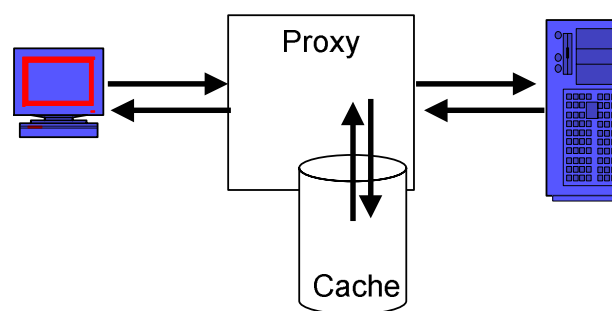


Figure 6 : Caching proxy

L'adjonction d'un cache implique une gestion de ce dernier. Premièrement, il faut évaluer les réponses qui peuvent ou non être cachées, et donc copiées dans l'espace de stockage. Certaines réponses ne sont pas cachables, et même pour les réponses qui le sont, il n'est

⁴ Situation privilégiée au centre d'une communauté.

pas toujours pertinent d'en retenir une copie. Cette décision s'appuie notamment sur des informations qui sont contenues dans la réponse et sont normalisées dans le protocole HTTP. Malheureusement, les directives de caching ne sont pas toujours correctes ou, le plus souvent, manquent dans l'en-tête des réponses. Ces dernières ont également une validité ; avant de servir une copie à un utilisateur, il faut s'assurer que la copie n'a pas expiré et de temps à autre s'assurer auprès du serveur d'origine qu'elle est toujours valide (mécanisme défini dans le protocole HTTP).

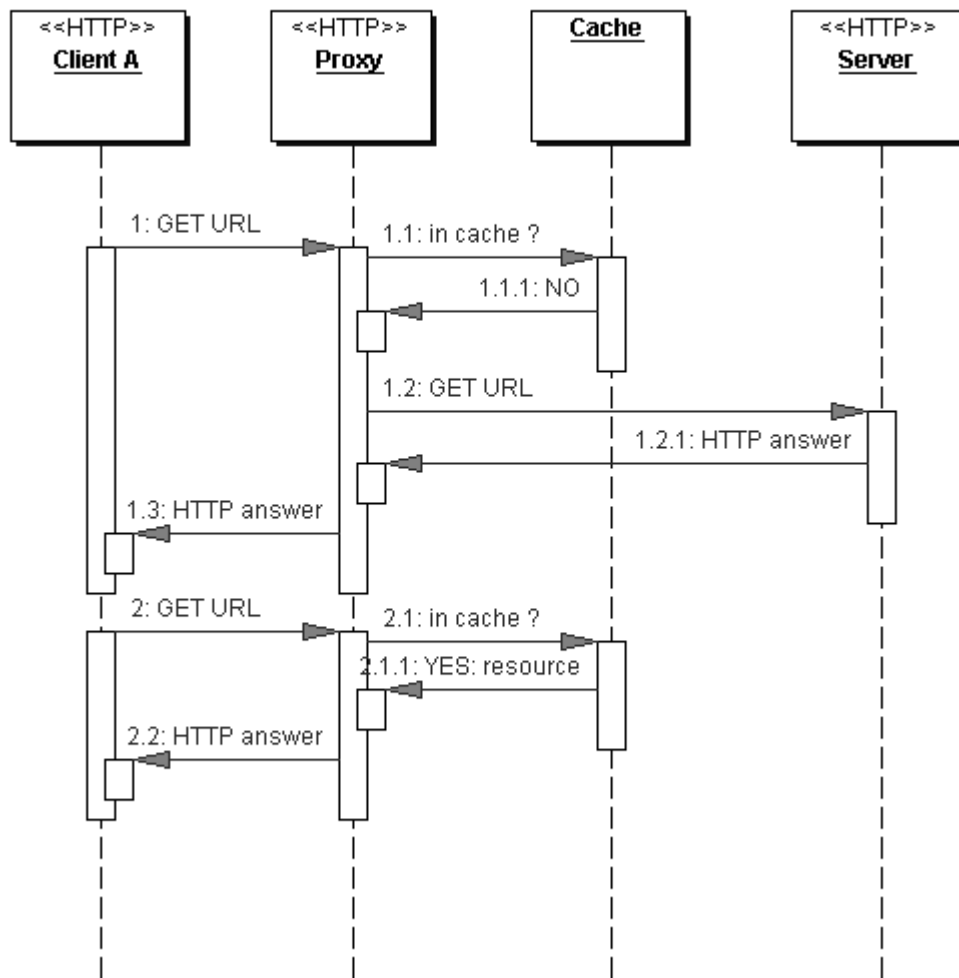


Figure 7 : Fonctionnement du caching

Il faut ensuite gérer un espace de stockage limité, alors que le volume des réponses en transit ne l'est pas. Toutes les réponses cachables ne peuvent donc pas être stockées dans le cache. Pour pallier ce problème, les proxys ont une politique de gestion du cache qui supprime, en fonction de critères qui font partie de la politique, des documents qui ne sont plus jugés pertinents pour être cachés. Nous détaillons plus loin les différentes politiques proposées pour la gestion du cache.

Enfin, à une échelle plus large, on trouve des caching proxys à différents niveaux du réseau qui forment donc une hiérarchie de caches. Par exemple, une organisation comme une université dispose d'un caching proxy tout comme l'organisme qui fournit des connexions Internet à différentes organisations. Ces proxys fonctionnent alors de manière collaborative selon deux schémas. Dans le premier cas, ils sont chaînés, c'est-à-

dire qu'ils jouent tour à tour le rôle de client pour le proxy suivant et le rôle de serveur pour le proxy précédent. On voit bien que cet enchaînement se fait sans aucune fonctionnalité additionnelle en s'appuyant simplement sur les capacités intrinsèques des proxys. Cet enchaînement entre les proxys s'établit dans une relation parent-enfant. Le proxy en amont de la requête tient le rôle d'enfant et le proxy en aval le rôle de parent. Elle se construit avec la configuration du proxy enfant en lui indiquant quel(s) parent(s) il peut contacter. L'ensemble des proxys ainsi liés construit une structure (quasi) en arbre.

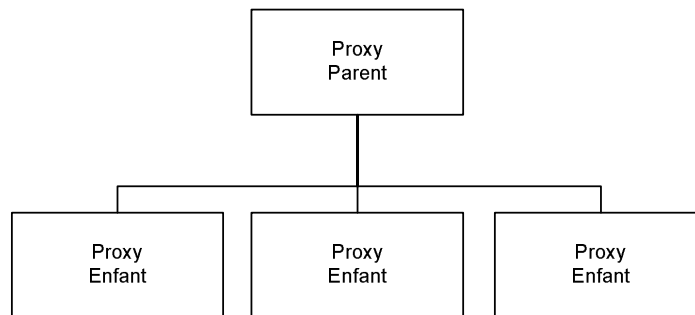


Figure 8 : Hiérarchie de proxys

Le deuxième schéma est plus compliqué et s'appuie sur différents modèles, mais il implique dans tous les cas la définition de nouveaux protocoles de communication. Il existe deux modèles de collaboration dans ce schéma, soit les proxys ont la possibilité de "consulter" des voisins, mais sans propagation de la requête, dans le but d'évaluer si la ressource sera plus rapidement disponible dans le voisinage qu'en la relayant vers l'origine ou vers un parent (le protocole ICP (Inter Cache Protocole) a été défini pour ce type de collaboration [31]), soit les proxys collaborent de manière proactive et fournissent à leurs voisins des réponses qu'ils ont eux-mêmes cachées sans que ces derniers en aient fait la demande. On parle alors de multicast [32].

Cette dernière problématique concernant la collaboration entre les caches n'a pas été abordée dans notre travail de recherche. Nous ne ferons que l'évoquer pour souligner que les solutions apportées restent valides dans le cas des relations parent-enfant, c'est-à-dire l'ensemble des transactions basées sur le protocole HTTP.

Notons cependant que les systèmes parents-enfants et ICP sont décrits plus avant dans [33] notamment en termes de performances. Une alternative pour la coopération entre les caches est également décrite dans [34], avec une proposition de nouveau protocole : Relais.

Enfin, on trouve dans [35, 36] des études approfondies sur les différences en performances des schémas envisagés. Il en ressort que le modèle parent-enfant présente des temps de connexion plus courts, mais des temps de transmission plus élevés qu'un modèle d'interrogation vers le voisinage en utilisant, par exemple, ICP. Ce constat induit que le premier modèle est plus performant pour des objets de petite taille. De plus, le deuxième modèle consomme, globalement, plus de ressources réseau. De même, par rapport au modèle parent-enfant, le modèle multicast consomme moins de ressources réseau et donne de meilleures performances en temps d'accès pour des documents qui changent très fréquemment, alors que c'est exactement le contraire pour des documents qui ne sont pas périodiquement modifiés.

3 Le protocole HTTP

Etant donné l'importance du protocole HTTP dans la problématique de notre travail, nous allons examiner, sans entrer dans tous les détails, l'ensemble du protocole. Il existe à l'heure actuelle deux versions du protocole: 1.0 et 1.1. La version 1.0 a été enregistrée sous forme de RFC 1996 et la version 1.1 existe déjà depuis 1997. Elles sont compatibles, ce qui veut dire qu'une application supportant HTTP 1.1 supportera également la version 1.0. Dans ce qui suit, nous allons examiner surtout les éléments qui co-existent dans les deux versions et nous signalerons certains ajouts qui ont été faits dans la version 1.1 en vue d'améliorer le fonctionnement des proxys. Cependant, sur ce dernier point, le protocole HTTP n'est de loin pas toujours supporté, et dans des transactions où un seul des agents ne supporte pas la version 1.1, tout ou partie de la transaction est alors ramenée à la version 1.0. Nous avons pris comme référence pour cette partie, essentiellement: [12, 37].

3.1 Historique

La première proposition [6] de Tim Berners-Lee décrit une architecture permettant notamment aux scientifiques de partager des documents et des notes en particulier destinées à la communauté scientifique du CERN. De ce rapport interne du CERN, ainsi que des premières spécifications [38], on retire l'idée générale suivante: un système HyperText distribué sur plusieurs serveurs, constitué de documents qui se référencent les uns les autres. Dans ces différentes notes, on retrouve ce qui va devenir le langage HTML à la base des documents du Web, le protocole http pour le transport de ceux-ci, et les liens qui relient les documents entre eux. Un point important à noter est qu'à ce premier stade de la réflexion, les liens entre les documents sont encore bidirectionnels (ex. A référence B / B est référencé par A) [39] et que l'on parle de la possibilité d'annoter les documents publiés [40]. Ce dernier point sera ensuite simplifié avec la normalisation des URL et la notion d'annotation disparaît du Web jusqu'à dernièrement avec l'apparition d'un des derniers projets de Tim Berners-Lee: le « Web semantic » [41] [42] qui revient sur les premières notions présentées dans [6]. Nous proposerons, dans la suite de notre travail, une solution intégrée au proxy concernant les annotations, la publication et le partage de celles-ci.

3.2 Introduction

Le protocole HTTP a été défini conjointement avec HTML (Hyper Text Meta Language), qui est un langage qui permet la construction de documents hypertextes. Ces deux spécifications constituent la base même du WWW (World Wide Web ou Web). Le principe du protocole HTTP est donc de permettre à un agent utilisateur d'accéder à des ressources disséminées sur Internet. Ceci suppose que l'utilisateur s'adresse à différents serveurs situés à différents endroits du réseau. Le protocole HTTP n'établit donc pas de connexion avec un serveur au-delà de l'accès à un document, mais il établit une connexion TCP à chaque requête pour une ressource, lit la réponse en retour et termine la connexion.

3.3 Requêtes et réponses

Le format d'une requête est donné ci-dessous (Figure 9). Les différents types de méthodes sont:

- GET qui retourne la ressource identifiée.
- HEAD qui retourne la même réponse que pour un GET, mais ne retourne pas la ressource elle-même, soit l'entité correspondante sans le contenu. Ce type de requête est utilisé pour obtenir des informations sur une ressource sans avoir à en télécharger le contenu. Elle est donc largement employée dans des mécanismes de validation pour le caching.
- POST qui permet le passage d'informations dans le corps de la requête. C'est le seul type de requête dont le corps contient des informations et qui exige une en-tête contenant l'information décrivant la taille de la requête.

Ainsi que les méthodes suivantes ajoutées dans la version 1.1

- OPTIONS similaire à HEAD, mais retourne des informations sur la communication avec le serveur.
- PUT permet de déposer une ressource sur le serveur.
- DELETE permet de supprimer une ressource.
- TRACE implémente une sorte d'écho. La requête est retournée au client.
- CONNECT permet à un client d'instancier un tunnel avec un proxy.

```
HTTP Request Syntax
```

```
<method><resource identifieur><HTTP version><crLf>  
[<Header> : <value>]<crLf>  
...  
[<Header> : <value>]<crLf>  
  blank line    <crLf>  
[entity body]
```

```
Example
```

```
GET /path/file.html HTTP/1.0  
Accept: text/html  
Accept: audio/x  
User-agent: MacWeb
```

Figure 9 : Format des requêtes

Le format des réponses est donné dans le tableau suivant (Figure 10). Contrairement aux requêtes, il n'existe donc qu'un seul type de réponse. Le code résultat décrit le résultat de la transaction. Par exemple, un code 404 signifie qu'il n'existe pas de ressources pour la requête formulée. On trouve dans les spécifications l'ensemble de ces codes et leur signification.

HTTP Request Syntax

```

<HTTP Version><result code>[<explanation>]<crlf>
[<Header> : <value>]<crlf>
...
[<Header> : <value>]<crlf>
    blank line    <crlf>
[entity body]

```

Example

```

HTTP/1.0 200 OK
Server: Apache/1.1
Mime_version: 1.0
Content_type: text/html
Content_length: 2000

<HTML>
<HEAD><TITLE> ....

```

Figure 10 : Format des réponses

3.4 Les directives

Comme nous l'avons vu dans la spécification, les requêtes et les réponses peuvent toutes deux contenir un certain nombre d'en-têtes. Une en-tête HTTP, appelée aussi directive http, afin de lever l'ambiguïté puisque en-tête désigne aussi bien tout ce qui précède le corps dans une requête ou une réponse, est composée d'un mot-clé suivi de paramètres. Il y a trois types de directives: celles qui s'appliquent aux requêtes, celles pour les réponses, et enfin celles qui donnent des informations sur le corps. Certaines directives s'appliquent aux réponses et aux requêtes. Celles qui donnent des informations sur le corps peuvent apparaître dans les réponses et dans les requêtes de type POST.

Directive	Requêtes	Réponses	Corps
Allow			X
Authorization	X		
Content-Encoding			X
Content-Length			X
Content-Type			X
Date	X	X	
Expires			X
From	X		
If-Modified-Since	X		
Last-Modified			X
Location		X	
MIME-Version	X	X	

Pragma	X	X
Referer	X	
Server		X
User-Agent	X	
WWW-authenticate		X

Figure 11 : Directives HTTP 1.0

Directive	Requêtes	Réponses	Corps
Cache-Control			X
Proxy-Authenticate	X		
Proxy-Authorization	X		
Range	X		

Figure 12 : Directives HTTP 1.1 qui concernent les proxys

3.5 Caching

Dans les tables ci-dessus (Figure 11, Figure 12), nous avons grisé les directives qui peuvent avoir une influence sur le caching et nous proposons ici quelques considérations concernant le protocole HTTP et le caching, notamment avec le support apporté par ces directives. Dans tous les cas, lorsqu'une ressource a expiré, le cache ne doit pas la fournir à l'utilisateur.

La directive d'expiration permet de déterminer de façon absolue si une ressource a expiré (Migros data du Web !).

Pour vérifier si une ressource est encore valide, il est possible de ne demander que des informations concernant la ressource sans en transférer le contenu (requête type **HEAD**).

Dans ce cas, on recevra dans la réponse une directive **Last-Modified** qu'il est possible de comparer avec la date de dernière modification de la ressource cachée. Dans le même ordre d'idée, il est possible de faire une requête conditionnelle avec la directive **If-Modified-Since** et le serveur peut ne pas envoyer la ressource dans la réponse.

La directive **Cache-Control** regroupe un sous-ensemble de directives destinées spécifiquement au contrôle du cache : autorisation de cacher, durée maximale de caching, interdiction de modifier, etc.

Le cache peut stocker des parties de ressources si une requête a été faite avec la directive **Range**. S'il reçoit ensuite une autre requête pour une autre partie de la même ressource, alors il peut combiner/compléter la copie cachée.

Il faut cependant bien comprendre que ces directives ne sont pas fréquemment utilisées ou alors de manière erronée. Souvent d'ailleurs pour s'assurer de pouvoir établir des statistiques d'utilisation.

Prenons un exemple en faisant `telnet ww.yahoo.com 80`, puis `GET / HTTP/1.0`. Le résultat illustre très bien le propos. L'en-tête de la réponse contient : **Cache-Control: private**. Ceci signifie donc que la page d'accueil de Yahoo est considérée comme privée en réponse à une requête qui ne contient aucune information d'authentification (!). Probablement que le serveur a été configuré pour envoyer systématiquement cette directive, mais malheureusement, selon les spécifications, cela interdit toute possibilité de caching pour les proxys.

Hormis pour le problème des statistiques d'accès que les proxys faussent, toutes ces directives de caching permettent en principe une gestion optimale dans tous les cas. Cependant, lors de la mise en place de services Web, cet aspect est presque toujours laissé de côté et l'accès aux directives HTTP n'est pas particulièrement aisé dans les serveurs à disposition.

3.6 *Rapport statistique*

Nous avons évoqué le problème des statistiques d'utilisation et souligné le fait que les proxys faussent ces évaluations. En effet, pour toutes les requêtes qui ne sont pas transmises au serveur d'origine, il n'est pas possible pour celui-ci de les comptabiliser. Ce problème induit les acteurs de l'Internet à entraver l'efficacité des proxys comme dans l'exemple de la page d'accueil de Yahoo. Les éditeurs de sites Web et les annonceurs qui mettent des publicités sur ces sites veulent pouvoir faire état du nombre d'accès à leurs documents. Ils s'appliquent alors à mettre en place de "l'écrasement de cache" (cache-busting) en interdisant le cache des objets au moyen des directives HTTP à disposition.

Ainsi, parmi les 50 sites les plus populaires, 26 ont mis en place de l'écrasement de cache sur leur page principale et 16 le font même sur leur page présentant les conditions d'utilisation, un document pourtant rigoureusement statique [43]. Une autre technique communément utilisée est l'insertion dans les documents de "Web bugs": une image gif transparente de 1x1 pixels. Ces Web bugs sont interdits de caching et permettent donc de révéler au serveur les accès au document tout en autorisant le cache de ce dernier. Cette dernière solution est moins coûteuse, mais elle génère tout de même un trafic qui repose non seulement sur la taille du Web bug (1 pixel !), mais également sur l'échange des entêtes de la requête et de la réponse.

Afin de pallier ce problème, un protocole permettant aux proxys de rapporter les statistiques d'utilisation au serveur d'origine a été mis en place [44]. Mais nous ne connaissons pas à ce jour d'implémentation qui supporte ce protocole, on trouve cependant dans [43] une excellente évaluation sur les performances qu'implique son utilisation.

4 **La Gestion du cache**

En stockant les ressources dans un voisinage plus proche du client, un caching proxy peut réduire le temps de réponse des requêtes, la charge des serveurs et le trafic réseau.

La gestion du cache dans l'utilisation d'un caching proxy est donc cruciale pour augmenter les performances apparentes du réseau et économiser l'utilisation des accès distants. C'est même le point central pour gagner en performance : éviter l'accès à des ressources distantes alors que tous les processus locaux sont comparativement de coût négligeable. Le reste du fonctionnement est donc relativement simple puisqu'il repose essentiellement sur l'implémentation client et serveur du protocole HTTP et a donc moins d'impact. Les gains en performances pour le cache peuvent s'établir sur deux paramètres : la rapidité d'accès en fonction des ressources où sont stockés les documents (mémoire, disques durs, etc.) et les performances apparentes pour l'utilisateur.

Deux principaux problèmes sont liés à la gestion du cache : d'une part on dispose toujours d'un espace de stockage limité par rapport à la taille du web (plus de 38 millions de sites référencés par netcraft (www.netcraft.com), et selon [45] plusieurs milliards de documents en 2002), d'autre part, les documents ont une durée de vie souvent très mal ou pas du tout établie selon les normes.

Pour ces deux raisons, le point le plus sensible des caching proxys est la mise en place d'une politique de remplacement des documents qui permette de maximiser les performances. Nous le détaillerons plus loin, mais plus la proportion de documents

servis à la requête de l'utilisateur sont extraits du cache par rapport à l'ensemble des documents requis, plus le caching proxy est performant.

Le caching n'est pas en soi une technique nouvelle et il a fait l'objet de recherches depuis des décennies. La définition la plus générale pour le cache serait : un agent autonome situé dans une hiérarchie de stockage (disques durs, mémoire centrale, etc.) qui gère l'espace cache. Il retire si nécessaire des objets de l'espace cache de manière réactive ou proactive, il insère des objets dans l'espace cache et il fournit des objets qui se trouvent dans cet espace. Il opère sur la base des points suivants:

- Limites de l'espace cache
- Politique(s) de remplacement
- Contrôle de la validité
- Règles d'admission

On parle de hiérarchie de stockage notamment en termes de performances. On utilise des caches partout où un stockage primaire présente un déficit en performances. On adjoint alors un cache pour pallier ce défaut. L'espace cache est alors un espace de stockage présentant de meilleures performances (mais généralement plus coûteux, donc limité !) qui permet de déplacer une partie de l'information dans une zone où l'accès est beaucoup plus performant. On trouve également des fonctions de caching lorsqu'un espace de stockage est coûteux et trop limité. On étend alors de manière virtuelle l'espace rapide en déplaçant des informations dans un espace plus lent mais plus large (optimisation des coûts).

L'espace de cache est limité, c'est une première contrainte incontournable qui implique donc que l'on ne peut pas tout cacher. Ceci s'exprime par:

$\sum s_i \leq S$, avec s_i la taille des documents dans le cache, et S la taille du cache.

Il faut donc mettre en place une gestion du cache qui s'assure d'abord que l'on ne dépasse pas cette limite et surtout que l'espace de cache soit utilisé de façon optimale. La gestion du cache doit donc, le cas échéant, supprimer des objets de l'espace cache pour y stocker d'autres objets. Cette opération se fait selon une politique de remplacement qui détermine, en fonction de critères prédéfinis et d'observations statistiques, les objets dont la présence dans l'espace cache contribue le moins aux gains de performances. Le cache peut agir soit de manière active, auquel cas il évalue régulièrement l'espace encore disponible et passé un seuil déterminé commence à éliminer des documents. Il peut aussi agir de manière réactive, auquel cas il supprime un objet lorsqu'il doit libérer de l'espace pour un autre objet. Le gestionnaire du cache qui manipule des copies des objets doit également veiller à ce que ces copies restent valides (identiques) par rapport aux originaux. Il peut également y avoir des règles d'admission qui autorisent ou excluent certains objets.

La principale préoccupation du caching est donc l'optimisation, soit des coûts, soit des performances. Dans les deux cas, une bonne gestion implique des gains en performances. On trouve des caches pour la mémoire centrale (pagination sur le disque), dans les bases de données (stockage en mémoire centrale d'ensembles d'informations pour des gains en performances), etc.

Dans le cas d'applications en réseau, le cache sert alors à optimiser la vitesse apparente de l'espace de stockage. On rapproche les informations de l'agent qui les consomme de façon à réduire les temps de transfert. On trouve de nombreux exemples dans le cas des bases de données réparties où des optimisations, par des déplacements judicieux d'ensemble d'informations, entraînent des gains considérables.

Dans le cadre du caching Internet, on peut relever les particularités suivantes : la taille variable des objets, l'absence de fragmentation des objets, des performances variables,

une distribution très large des documents (**World Wide Web** !), un nombre également très important d'utilisateurs [46] et une gestion de la cohérence particulière.

La disparité en taille des documents disponibles sur Internet est considérable : de quelques octets pour un document, à plusieurs Gigas pour des ressources contenant du multimédia. Cette hétérogénéité influe nettement sur la gestion du cache par comparaison à un système de pagination de la mémoire où les pages sont justement de taille fixe. Certains objets seront donc beaucoup plus coûteux à cacher que d'autres. La version 1.1 du protocole supporte la fragmentation des objets (transferts par parties), mais l'utilisation du Web implique presque systématiquement l'accès à des ressources complètes.

Les performances du stockage d'origine des ressources sont variables en terme de vitesse apparente, que ce soit le temps d'accès ou le temps de téléchargement. On tirera, en fonction de ces variations, beaucoup plus de bénéfice à cacher certains objets.

Le nombre d'agents utilisateurs est également élevé. Les caches Web sont souvent partagés par des milliers d'utilisateurs. Contrairement à la pagination mémoire ou une, voire quelques applications, consomment de l'espace mémoire, on se retrouve avec des utilisations très variées.

Enfin, la cohérence pour les documents du Web n'est pas aussi cruciale que dans la plupart des autres cas. Les ressources sont en effet rarement destinées à des agents informatiques mais à des personnes. De plus, au moins pour tous les documents statiques, les mises à jour sont souvent rares et unilatérales (mise à jour côté serveur uniquement). Il existe cependant des directives d'invalidation, mais elles sont embarquées dans les ressources distribuées. Cela veut dire qu'il n'est pas possible pour les serveurs d'origine de forcer la revalidation d'un objet caché. Les copies d'objets sont nombreuses et non répertoriées. Le problème lié à l'invalidation des documents procède aussi du protocole http, avec une gestion correcte des en-têtes. En effet, on constate qu'il est rarement fait bon usage des en-têtes http pour spécifier la durée de vie des documents (voir exemple de Yahoo!). La problématique liée au protocole est largement décrite dans [21].

En résumé, la gestion du caching dans le cadre du Web n'a pas d'application équivalente. C'est pourquoi, bien que la technique du caching soit éprouvée depuis longtemps, l'expérience acquise dans les autres domaines n'est que peu applicable dans ces cas. Enfin, de nombreux paramètres rentrent en ligne de compte pour approcher une utilisation optimale, raison pour laquelle de nombreuses politiques de gestion des caches Internet existent concurremment à ce jour.

Les caches Internet appliquent une gestion proactive : lorsque l'on s'approche d'une certaine limite, ils purgent le cache d'un certain nombre d'objets pour pouvoir continuer à en stocker de nouveaux. Le principe étant de garder les objets qui seront les plus rentables au niveau des performances, selon une métrique que nous décrivons ci-dessous, et donc d'extrapoler à partir des informations disponibles quels sont les documents auxquels on pourra accéder, ceci avec des algorithmes réalistes qui ne demandent pas trop de temps de calcul. De très nombreux algorithmes ont été proposés à ce jour.

Avant de présenter un tour d'horizon des techniques existantes, nous allons commencer par exposer quelques métriques pour évaluer les gains que peuvent apporter les différentes gestions du cache.

4.1 Dimensions

Comme nous l'avons signalé dans l'introduction, nous nous intéressons particulièrement à la perception des dimensions par le proxy. Avant d'étendre cette perception à de nouvelles dimensions dans les chapitres suivants, nous rapportons ici celles qui sont prises en compte dans les politiques de gestion du cache existantes.

Nous avons d'abord les dimensions liées à l'objet:

- La taille
- Le type

Nous avons ensuite les caractéristiques qui dépendent de sa localisation (serveur d'origine):

- Le temps de téléchargement
- Le temps d'accès (latency)

Et nous avons enfin les données statistiques qui varient en fonction du comportement des utilisateurs:

- Nombre d'accès (références)
- Date/Temps du dernier accès

4.2 Métriques pour la mesure des performances

Le but du caching proxy est de servir localement des documents contenus dans le cache et ainsi de s'économiser la requête au serveur d'origine dans un but d'optimisation. Le gain peut se traduire par une vitesse apparente accrue pour l'agent utilisateur ou une économie de la consommation réseau. Comme nous l'avons déjà évoqué précédemment, les critères qui entrent en ligne de compte dans le cadre du caching Internet sont nombreux. Ils se basent sur autant de dimensions qui peuvent être perçues par le proxy. Nous proposons ici quelques métriques pour l'évaluation des gains possibles en fonction de différentes dimensions. Nous parlerons d'abord des trois principales métriques qui dénotent des fonctions de rendement. On trouve également le concept de coûts ou de bénéfices dans d'autres travaux.

4.2.1 Rendement

Pour mesurer le rendement du processus, nous trouvons les fonctions suivantes.

Définition 1: le hit-rate est le pourcentage de requêtes dont le résultat est un document issu du cache sur le nombre total de documents retournés par le proxy.

$$\eta_{hit} = \frac{\sum D_{i,cache}}{\sum D_{i,out}}$$
 avec $D_{i,cache}$ les documents qui sont servis depuis le cache et $D_{i,out}$ l'ensemble des documents que le proxy délivre à des clients.

Définition 2: le bit-rate est le pourcentage d'octets qui sont extraits du cache sur le total des octets qui sortent du proxy en direction des clients.

$$\eta_{bit} = \frac{\sum o_{i,cache}}{\sum o_{i,out}}$$
 avec $o_{i,cache}$ les octets qui sont servis depuis le cache et $o_{i,out}$ l'ensemble des octets que le proxy délivre à des clients.

Ces deux fonctions représentent le rendement du proxy selon deux points de vue différents (décrites dans [47] comme le FAULT model et le BIT model). Le hit-rate représente le taux de documents sortis du cache. Le bit-rate représente l'économie de trafic réseau sur l'extérieur réalisée par une communauté utilisant un caching proxy. Le bit-rate est équivalent au hit-rate pondéré par le poids des documents. Cette deuxième mesure est donc en général plus pertinente. On utilise cependant encore souvent le hit-rate en regard des politiques de cache qui procèdent par éviction de documents, souvent d'ailleurs sans tenir compte de la taille de ces derniers. En fait, on comprend bien que

dans le cas où l'on cache par exemple deux très gros documents au détriment de cinq cents petits, on aura de bien meilleures performances au niveau du bit-rate et que par contre les résultats basés sur le hit-rate seront catastrophiques. Il faut d'ailleurs souvent nuancer les résultats affichés dans les politiques proposées par le fait qu'ils peuvent être excellents dans une mesure, mais bien moindres selon une autre métrique.

On trouve une autre métrique dans [48] qui prend en compte le gain en temps pour l'utilisateur et définit le rendement.

Définition 3: Le delay saving ratio (DSR) est le temps gagné par l'usage du cache pour l'utilisateur.

Qui se calcule avec:

$$DSR = \frac{\sum_i (d_i \cdot h_i - c_i \cdot v_i)}{\sum_i d_i \cdot f_i}$$

avec d_i, h_i, c_i, v_i, f_i , respectivement : le temps de téléchargement, le nombre de hits dans le cache, la coût pour valider le document auprès du serveur d'origine, le nombre de requêtes de validation et le nombre total de requêtes, pour un document i

On pourra également définir un rendement simplifié en définissant d'abord:

Définition 4: L'intervalle de temps visible pour le proxy associé à une requête désigne l'intervalle de temps qui s'écoule entre le moment où il a reçu l'entier de la requête adressée par un client et le moment où il ferme la connexion pour cette même requête.

$$\eta_{time} = \frac{\overline{\Delta f_{cache}}}{\overline{\Delta f_{out}}}$$

avec $\overline{\Delta f_{cache}}, \overline{\Delta f_{out}}$, respectivement l'intervalle de temps moyen visible par le proxy pour délivrer un objet du cache et l'intervalle de temps moyen visible par le proxy pour un objet résultant d'une requête transmise au serveur d'origine.

Certains [8, 46] par contre critiquent l'utilisation de cette dimension dans l'évaluation des algorithmes et tombent tout naturellement sur des résultats mettant en avant le caching de nombreux objets de petite taille. Comme le gain en temps de chargement dépend de la taille, ces résultats semblent un peu biaisés.

4.2.2 Coût/Bénéfice

On définit le coût ou le bénéfice d'un algorithme sur un ensemble de documents comme le gain réalisé en fonction d'un des calculs de rendement présentés ci-dessus (dans [3, 5]).

Dans [49] on trouve une définition du coût: $C \cdot P_r$ avec P_r la probabilité que le document soit de nouveau accédé et C selon plusieurs critères:

- les connexions (en général $C = 11$ puisqu'il faut une connexion par ressource !)
- le trafic : $C = \text{taille} + \text{en-tête}$, c'est-à-dire la consommation réseau
- le temps : C est proportionnelle à la durée de téléchargement de la ressource

Dans [50], on définit encore le bénéfice pour un document D_i donné qui se trouve dans le cache: $bénéfice(D_i) = \frac{\lambda_i \cdot d_i}{s_i}$, avec λ_i le taux estimé de référence au document, d_i le temps moyen de téléchargement et s_i la taille du document.

Ces différentes métriques se justifient parce que le gain dépend du point de vue. Ainsi, pour les différents algorithmes qui concernent la politique de gestion du cache que nous présentons plus loin, les résultats sont plus ou moins bons en fonction de la métrique utilisée.

5 Etat de l'art et classification des politiques de gestion

Nous avons défini les dimensions accessibles pour le gestionnaire de cache et les différentes métriques à disposition. Nous avons vu que pour l'un et l'autre, les possibilités étaient nombreuses. Ceci explique le fait qu'il existe de nombreuses politiques de gestion qui prennent en compte différents paramètres ou combinaisons de paramètres et visent à fournir une optimisation selon telle ou telle métrique.

Nous proposons de faire ici un tour d'horizon des différentes politiques à disposition et pour cela nous définissons tout d'abord un ordre qui désigne le nombre de paramètres pris en compte.

5.1 Politique simple d'ordre 1

Par simple, nous désignons l'usage de paramètres intrinsèques au document lui-même, par opposition à des paramètres induits par des opérations, par exemple d'accès à ce document. Ces paramètres ne dépendent donc que du document lui-même, sorti de tout contexte.

5.1.1 *Size*

Il s'agit d'une politique d'éviction qui ne prend en compte que la taille des documents. Cette information, selon la définition du protocole, devrait être présente dans les en-têtes des réponses http (voir [12] pour plus de détails) et peut de toute façon très facilement être déterminée à la volée. L'algorithme cherche simplement à maximiser l'espace gagné lors de l'éviction de documents en éliminant les plus volumineux.

De prime abord absurde, cette politique présente plusieurs avantages : la facilité de mise en place, la rapidité et l'efficacité pour retrouver de l'espace dans le cache. De plus, elle donne des résultats surprenants selon [4, 51]. Mais au fil du temps (probablement plus que les tests effectués !) on risque de se trouver avec un cache complètement pollué par des objets de petite taille.

5.1.2 *Autres paramètres*

Il existe d'autres paramètres intrinsèques aux documents comme le type, la date de création ou de modification. Le type est parfois utilisé pour filtrer les documents qui entrent dans le cache, mais nous ne connaissons pas de politique qui en tienne compte au niveau de l'indexation. De même, la date de création ou de modification entrent le processus d'invalidation, mais pas d'éviction.

Il existe d'autres paramètres qui ne sont pas intrinsèques au document lui-même, mais sont exprimés par des directives qui accompagnent le documents sitôt que celui-ci est transporté par le protocole HTTP. Ce sont, par exemple, les directives proxy (`no-cache`, `expire`, etc.) que peut émettre le serveur d'origine. Ces paramètres influent soit sur la décision de cacher un document, soit sur l'éviction. D'une part, la gestion de

cache impliquée est largement décrite dans [12], d'autre part, on constate que malheureusement à ce jour ces directives sont très mal utilisées (voir ci-dessus l'exemple de Yahoo).

5.2 Politiques statistiques d'ordre 1

Les politiques statistiques prennent en compte l'utilisation, en terme d'accès aux documents, que l'on peut observer du point de vue du proxy. Elles se basent donc sur le comportement des utilisateurs qui partagent le proxy et non plus sur les informations intrinsèques au document lui-même ou faisant partie des directives du protocole HTTP. Nous avons exclu la politique FIFO (First In First Out) parce qu'elle nous semblait d'évidence inadéquate.

5.2.1 LRU - Last Recently Used

Pour tout objet caché, on stocke et on met à jour à chaque nouvel accès la date d'accès. La politique d'éviction consiste alors à éliminer les objets auxquels on n'a pas accédé depuis le plus longtemps.

On trouve aussi une variante: LRU-Threshold dans [22], qui consiste à ne pas cacher les objets dépassant une certaine taille.

5.2.2 LFU - Least Frequently Used

Pour tout objet caché, on stocke et on met à jour un compteur du nombre d'accès à cet objet. La politique d'éviction consiste alors à éliminer les objets auxquels on accède le moins fréquemment.

On trouve la description et les résultats en performances de ces deux algorithmes dans [4, 15, 22].

5.2.3 Page load delay

Dans [2], il est proposé d'éliminer le document qui présente le plus faible temps de transfert. Cette politique s'attache à maximiser le Delay Saving Ratio.

5.2.4 EXP1

Cette politique présentée dans [5] est en fait une combinaison de LRU et LFU. Le principe est de modérer les effets de l'algorithme LRU en prenant en compte la fréquence d'utilisation d'un document.

5.2.5 LRV - Lowest Relative Value (LRV)

Cette politique [3, 49] est basée sur l'évaluation de l'utilité de conserver un document dans le cache. Le document avec la plus petite évaluation est supprimé. Le calcul de l'utilité se base sur l'historique du cache.

5.3 Politique d'ordre n

Ces politiques prennent en compte n paramètres et les combinent.

5.3.1 Log-Size

Log-size ([4, 22]) propose de partitionner l'espace des objets du cache selon une fonction $\log(\text{size})$ et d'évincer des ressources dans chaque partition en appliquant LRU. Le principe étant de favoriser la sortie des documents de plus grande taille.

5.3.2 Greedy-Dual-Size (with frequency): GD-Size et GDSF

Cet algorithme, décrit dans [3, 46, 51], se base sur l'évaluation du coût défini par:

$K_i = C_i/S_i + L$, avec C_i le coût associé à l'acquisition de l'objet i dans le cache, S_i la taille de l'objet et L un facteur qui croît avec le temps.

Cette politique donne d'excellents résultats comme le montre l'analyse des performances [52].

5.3.3 *Least Frequently Used with dynamic Aging (LFU-DA)*

Cet algorithme décrit également dans [51] est très similaire au Greedy-Dual Size, mais cette fois, au lieu de prendre en compte la taille de l'objet, on considère la fréquence des accès.

5.3.4 *Hyper-G*

Cette politique combine LFU avec SIZE, décrit dans [4, 53].

5.3.5 *LNC-R-W3-U*

Cette politique décrite dans [48, 50] vise à optimiser le coût de téléchargement de l'objet dans le cache ainsi que la fréquence à laquelle les documents sont mis à jour (du côté du serveur d'origine). Ainsi, des documents qui se modifient régulièrement (par exemple des sites de news) sont plus volontiers écartés.

5.4 *Politique à étages : caches virtuels*

Dans ce genre de politique décrite dans [51, 53], on n'évince pas directement les documents, mais on les déplace dans une autre zone de stockage avec une autre politique d'éviction. On peut ainsi enchaîner les étages.

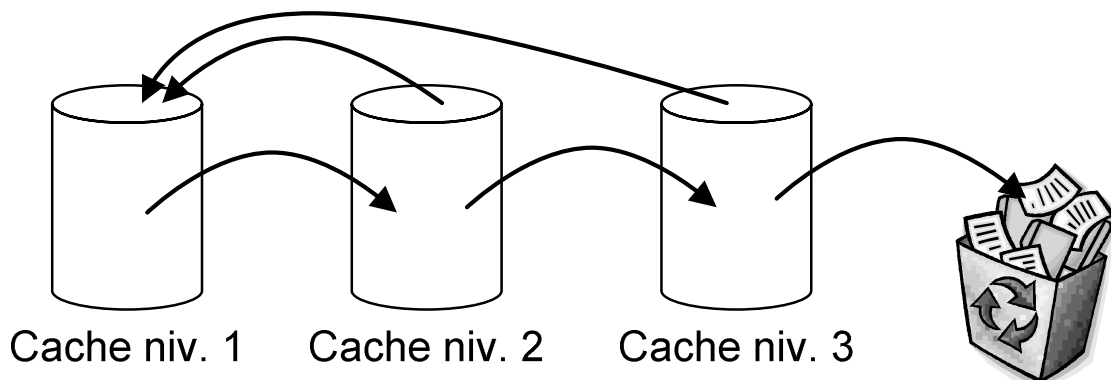


Figure 13 : Cache à étages

Le principe est que les documents sont cachés dans un premier niveau de cache. Lorsque, selon les critères de la politique de gestion, ils doivent être supprimés, ils sont alors transférés dans un deuxième niveau, puis dans un troisième, et ainsi de suite en fonction du nombre de niveau existants, jusqu'à une éventuelle suppression complète (Figure 13). Cette disposition permet ainsi de combiner plusieurs politiques de gestion différentes et donc, le cas échéant, de bénéficier des différents avantages.

5.5 *Hardware*

La dernière caractéristique à prendre en compte pour la gestion du cache se situe au niveau du hardware. En effet, l'architecture physique du cache peut influencer sur les performances : taille de la mémoire à disposition, caractéristiques du disque dur. Pour ce dernier point, on trouve dans [54] une proposition pour la mise en place de la politique LRU, tout en prenant en compte la structure physique de l'espace de caching sur le

disque. Nous n'avons pas considéré plus loin cet aspect dans le reste de notre travail, bien que de nombreux développements soient encore possibles en intégrant les travaux faits dans le domaine de l'optimisation du stockage.

5.6 Synthèse

De manière générale, si les nouvelles techniques proposées permettent de gagner un peu en performance, en pratique, aucune technique ne dépasse le système LRU de beaucoup, alors que ce dernier est relativement facile et peu coûteux à mettre en place. La plupart des propositions citées sont accompagnées d'évaluations statistiques prometteuses (le plus souvent des simulations). Mais, de manière générale, les avantages qu'une nouvelle politique de gestion apporte dans certains cas s'accompagnent de faiblesses sous certaines conditions. Ainsi, telle politique donne d'excellents résultats selon telle métrique mais pas du tout selon telle autre. On constate d'ailleurs très bien dans la pratique que si beaucoup de recherches théoriques sont présentées, elles n'ont que peu d'impact sur les réalisations pratiques au point de se demander si ces recherches ne sont pas vaines⁵. Il existe également des travaux comparatifs qui synthétisent dans le détail les différences que l'on peut attendre des techniques proposées [3, 46, 51].

Notre point de vue est qu'une nouvelle méthode d'indexation devrait s'accompagner de nouvelles fonctionnalités pour réellement susciter un intérêt. C'est ce qui a motivé nos travaux pour aboutir à la solution présentée dans le chapitre suivant.

5.6.1 *Persistence des index et aberrations*

Toutes les implémentations de caching proxys, à ce jour, ne stockent dans l'index des objets du cache que les objets qui s'y trouvent encore. A partir du moment où un objet est évincé, aucune information ne subsiste. Si dans le cas des objets invalidés, cette politique est parfaitement correcte, elle peut par contre aboutir à des effets de bord pernicious, notamment dans le cas d'une politique LFU. Prenons par exemple le cas de cnn.com qui archive tous les articles avec un identifiant unique. Un document d'actualité connaîtra une fréquentation très élevée le jour même, puis celle-ci va très fortement décroître pour tendre vers une valeur nulle. Imaginons maintenant un autre document auquel on accède à intervalle régulier. Soit T la période où l'on applique la politique d'éviction du cache et t la période d'accès à ce document avec $t > T$. On voit bien qu'après $x \cdot T, \forall x$, c'est toujours le document avec une forte activité au début qui l'emporte. Le résultat reste fortement probable si l'on pose des variables aléatoires pour les deux intervalles.

Un raisonnement similaire permet de trouver des points de distorsion pour la politique LRU.

Une bonne solution pour éviter les aberrations mentionnées est de conserver un index des objets au-delà de leur éviction du cache. On élargit ainsi la collection des statistiques. Peu de travaux s'engagent sur cette voie, on trouve cependant une solution d'index partiellement persistant dans [4], mais nous pensons que de systématiser la persistance des références dans l'index permettrait également de fournir des services de recherche dans la mesure où cet index atteint une certaine complétude. L'ancêtre de Squid, Harvest, proposait donc une indexation plus développée. Cependant, cette caractéristique a été abandonnée dans Squid avec la suppression des outils adjoints au système de caching.

De plus, un index complet permettrait d'envisager de collecter des statistiques sur le cycle de vie de l'objet (TTL, Time To Live) avec des informations comme le temps moyen

⁵ Question d'ailleurs débattue lors du WCW'2001 à Boston lors de la table ronde du 20 juin intitulée : « Publish no more papers on Web Cache replacement policies ».

entre modifications (en étendant donc l'idée de LNC-R-W3-U ou Least Frequently Used with dynamic Aging).

6 Prefetching

Le prefetching désigne l'opération d'anticiper le chargement d'informations. Dans le cadre des proxys Internet, elle s'applique à deux endroits : le pré-chargement d'informations de l'espace cache vers la mémoire, une requête anticipée adressée au serveur d'origine.

Le comportement des proxys Internet décrit jusqu'ici est essentiellement réactif. Lorsqu'une requête est adressée, l'agent de gestion du cache retrouve l'objet dans l'espace cache le cas échéant, ou évalue la possibilité de la ressource. Si l'espace cache dépasse un certain seuil, alors il lance une procédure de suppression d'objets.

En anticipant les requêtes des utilisateurs, on peut cependant gagner en performance. L'espace cache est généralement stocké sur un disque dur plus lent, mais de plus grande taille que la mémoire centrale. Les ressources qui sont déjà en mémoire centrale sont donc plus rapidement accessibles, d'où l'intérêt d'y pré-charger des ressources dans la mesure où l'on peut prédire qu'elles vont prochainement être requises. Dans le même ordre d'idée, si l'on peut prédire que telle ou telle ressource va prochainement faire l'objet d'une requête mais qu'elle ne se trouve pas actuellement dans l'espace cache, alors on pourra induire un gain en performances au niveau de la vitesse apparente de chargement, si l'on télécharge cette ressource à l'avance.

Il faut alors installer dans le proxy un processus proactif reposant sur des évaluations prédictives. La problématique est complexe parce que les agents qui rentrent en jeu sont particulièrement nombreux : tout d'abord des millions de serveurs et encore plus de ressources différentes du côté des sources, ainsi que de nombreux utilisateurs du côté des consommateurs.

Il existe deux modèles pour le prefetching : l'un contrôlé par le proxy (ou le client), l'autre contrôlé par le serveur [55]. Dans un environnement contrôlé par le serveur, c'est ce dernier qui prend les décisions de prefetching en se basant sur les statistiques qu'il possède. Bien qu'elles soient les plus fiables, cela induit de nombreuses communications supplémentaires entre le serveur et le proxy. Dans le contrôle du prefetching par le proxy, c'est ce dernier qui, en se basant sur ses propres statistiques d'utilisation initie le prefetching.

Une première catégorie d'algorithmes de prefetching se base sur des prédictions à court terme. On élabore des statistiques mesurant les probabilités d'enchaînement d'un document vers un autre groupe de documents. Pour cela, on trouve dans [56] une proposition de protocole basée sur le protocole HTTP pour échanger des informations entre le client et le serveur. Par exemple, si le client émet une requête pour un document P suivie d'une requête sur le document Q, alors il envoie un rapport au serveur pour indiquer cette séquence. Le serveur collectionne ainsi les rapports pour établir des statistiques de transition.

Et enfin, on trouve dans [55] une proposition de coordination entre le proxy et les serveurs pour limiter la surcharge impliquée par l'échange de rapports pour pouvoir établir des statistiques d'utilisation.

6.1 Chaînes de Markov

On utilise alors un modèle statistique de Markov pour évaluer les probabilités d'enchaînement. Cet algorithme appelé PPM (Prediction by Partial Match) est décrit dans [57] ou encore dans [58], dans un contexte de stockage hiérarchisé. La proposition faite dans [56] propose également de prendre en compte la directive HTTP `referer` (Figure

11) en signalant cependant que cette dernière n'est pas toujours présente en fonction des agents utilisateurs. On trouve enfin dans [56, 59] une étude mettant en place ces techniques pour des cas de bande passante réseau réduite.

6.2 Volumes

Dans [9, 10] on trouve le concept de volume pour regrouper des documents qui sont liés par des probabilités fortes d'enchaînement.

Définition 5: Un volume est un groupe de ressources qui sont statistiquement liées par des accès simultanés, quasi simultanés ou dans un intervalle limité.

L'élaboration de ces volumes peut se faire selon différents critères:

- Les statistiques d'utilisation (similaire à la proposition précédente, implique des rapports fournis par les clients)
- Le "voisinage" des documents dans la structure de fichiers du serveur (ou similarité des URL si l'on n'a pas la visibilité de la structure de fichiers)
- "L'utilité" de rajouter un document dans le volume (algorithme greedy). Le calcul de l'utilité dépend du critère d'optimisation
- Par comparaison : si le document se trouve dans un volume V_1 , on regarde si comparativement le gain serait meilleur en le mettant dans le volume V_2
- Les fréquences de mises à jour de l'objet sur le serveur d'origine [60].

Toutes ces démarches sont transposables au cas des proxys Internet. Cependant, pour certaines situations, on comprend bien que le proxy manque de données pour son évaluation et ne peut surtout pas prédire l'accès à des ressources qui ne se trouvent pas dans l'espace cache. C'est le cas notamment pour la construction de volumes basée sur la structure de fichiers.

6.3 Synthèse

La plupart des algorithmes de prédiction s'appuient donc sur les statistiques de transition d'une ressource à une autre. Mais le premier problème lié aux techniques proposées réside dans l'augmentation du trafic généré induit par les rapports échangés. C'est le même problème que pour les rapports de statistiques évoqués au paragraphe 3.6 p.25, également développé dans [43]. Cette problématique est liée au manque de vue d'ensemble lié au rôle d'intermédiaire du proxy et peut être compensée par la perceptivité des données disponibles. On voit alors que l'extension de la perceptivité du proxy amènerait des améliorations également dans ce domaine.

Les chaînes de Markov posent un certain nombre de problèmes de modélisation. Dans [56], on souligne d'abord la distorsion qu'introduisent les objets "inclus" (notamment les images). Pour résoudre ce problème, il est proposé de regrouper les ressources référencées dans un certain intervalle de temps. Or, ceci présuppose le concept de session qui n'est pourtant pas disponible (!). Et la définition d'un intervalle de temps nous semble très aléatoire puisque dépendante des performances du réseau.

Les simplifications proposées dans [57] ne tiennent pas du tout compte de ce problème, ce qui amène [60] à écarter tout simplement la modélisation par chaînes de Markov pour l'anticipation à long terme. Dans [7], il est également supposé que les chemins anticipés sont relativement courts.

7 Statistiques d'utilisation

Comme l'évaluation du bénéfice/coût de stocker une ressource dans l'espace de cache ou alors le prefetching de cet objet dépendent en grande partie des statistiques d'utilisation, il est important de connaître une modélisation de ces statistiques.

Il a été démontré dans plusieurs études [61, 62] que la probabilité qu'on accède à un document était liée à sa popularité par une loi de Zipf. Soit le i ème document par rang de popularité a une probabilité d'accès inversement proportionnelle à i . Dans le cadre du Web, cette popularité suit une courbe de la forme: $\frac{1}{i^\alpha}$.

Donc, pour un ensemble N de pages Web, la probabilité relative pour le document de rang i on obtient:

$$p_i = \frac{C}{i^\alpha}, \text{ avec } C = \frac{1}{\sum_{k=0}^N \left(\frac{1}{k^\alpha}\right)}$$

Cette loi de distribution a été vérifiée dans de nombreux cas en étudiant les logs de proxys, mais toujours sur des ensembles très importants de requêtes analysées. L'estimation est plus que satisfaisante, mais on connaît par contre moins bien ses limites, notamment la taille minimale pour qu'elle se vérifie encore.

8 Conclusion

Le protocole HTTP implique des limitations contraignantes, à commencer par la durée des connexions confinée à une transaction requête-réponse. Une solution consiste à proposer un nouveau paradigme. C'est ce qui est tenté au travers des CDN, où l'efficacité de la distribution est privilégiée. Cependant, on perd immédiatement l'universalité du Web même en reprenant une partie des spécifications. De plus, les architectures CDN supposent une infrastructure centrale contrôlée. Même si cette infrastructure est distribuée à l'échelle mondiale comme dans le cas d'Akamai qui a réussi à s'implanter chez une large proportion des acteurs de l'Internet, on ne peut pas indépendamment "s'installer" un proxy Akamai.

Or, les proxys ne demandent pas de modification dans la conception des infrastructures existantes, ils s'intègrent parfaitement et sans impact dans l'environnement Web. Et comme nous l'avons souligné, ils endossent alors de manière transparente un rôle central. De plus, cette intégration dans un protocole universalisé permet leur intégration dans des applications singulières sans aménagements particuliers non plus.

Une des fonctions centrales du proxy, le caching, pose un problème particulier d'optimisation. Par rapport au caching classique, les proxys et les applications Web ne posent pas des contraintes aussi sévères en matière de maintien de la cohérence. Les algorithmes d'invalidation sont donc beaucoup plus souples. Mais cette souplesse tolérée induit malheureusement souvent la mise en place de directives qui excluent le caching par des agents intermédiaires d'un nombre sans cesse croissant de ressources. Le protocole HTTP 1.1 apporte des solutions élégantes à ces différents problèmes, mais on constate qu'elles ne sont que peu ou mal exploitées.

En plus du maintien de la cohérence du cache se pose le problème de l'optimisation. De nombreux paramètres entrent en ligne de compte et il existe différentes métriques pour mesurer les performances. De la diversité du Web ont émergé un nombre important de propositions pour la politique de gestion du cache. Nous avons vu que ces différentes solutions proposent des gains nuancés et qu'à l'heure actuelle aucune politique ne l'emporte suffisamment pour s'être imposée comme standard. Nous considérons que,

pour qu'une proposition de gestion du cache suscite de l'intérêt, il faudrait qu'elle apporte plus qu'un léger gain en performance dans certaines circonstances.

Le prefetching permet également des améliorations. Comme nous l'avons vu, cette technique repose sur la capacité à prédire les requêtes futures probables. Plusieurs algorithmes sont proposés, mais la plupart induisent une augmentation du trafic avec des échanges de rapports statistiques. Un deuxième problème réside dans la modélisation et l'exploitation des données statistiques, d'où l'intérêt d'étendre la perceptivité du proxy. L'utilisation des chaînes de Markov semble s'adapter relativement bien à la navigation sur le Web, mais les modèles proposés présentent cependant des lacunes et soulignent le problème du manque de sessions.

Dans la suite, nos recherches visent à profiter de la situation centrale du cache en appuyant la mise en place de services sur le plésiocentrisme des proxys et en élaborant des techniques qui préservent l'intégration transparente dans des infrastructures basées plus spécifiquement sur le protocole HTTP.

INDEXATION SEMANTIQUE

Proposition d'indexation des ressources dans le cache en prenant en compte de nouvelles dimensions simplement perceptibles

1 Introduction

Nous avons présenté les différentes dimensions qui sont perçues par les proxys Internet dans leur fonction de caching. Nous avons également parcouru les différentes politiques de gestion du cache qui se basent sur ces dimensions. Celles-ci donnent des résultats variables, mais affichent des gains en performances nuancés. Dans ce chapitre, nous allons proposer une nouvelle méthode d'indexation des documents en prenant en compte de nouvelles dimensions.

Toutes les politiques de gestion que nous avons présentées reposent sur un ensemble fini de paramètres. Même si les algorithmes évoluent en intégrant toujours plus de paramètres dans des formules plus complexes, l'espace décrit, en terme de dimensions, n'évolue pas. Nous allons ici proposer d'introduire de nouvelles dimensions et des algorithmes adaptés à cet espace étendu.

L'objectif n'est pas seulement de proposer une politique de cache plus performante, mais surtout d'étendre la perception du cache, comme nous l'avons annoncé en introduction. Nous montrons en effet comment, à partir de ces nouvelles dimensions, de nouveaux services peuvent être envisagés.

Nous partons du principe que les documents Web ont une localisation porteuse de sémantique. Cette localisation s'appuie sur deux mesures : l'une, absolue, est exprimée par l'URL (Uniform Resource Locator) qui identifie la ressource ; et l'autre, relative, qui repose sur le voisinage du documents avec d'autres documents, est exprimée par les liens entre ceux-ci.

Les URL, dans leur spécification, n'ont pas de sémantique normalisée. Cependant, nous allons montrer que les éléments qui les constituent peuvent être porteurs d'un certain nombre d'informations concernant la localisation du document sur la toile. De plus, ces informations peuvent traduire un environnement culturel, un domaine d'activité, etc.

Les liens entre les documents, et surtout la fréquentation de ces liens par les utilisateurs qui passent d'un document à l'autre, décrivent aussi une topologie du Web. On peut révéler des distances relatives qui regroupent les documents en extrayant une sémantique de la fréquentation de ces chemins qui relient les différents points du Web.

En ayant extrait de ces différents éléments une sémantique qui décrit une géographie du Web ou un espace Web composé de positions, de distances relatives et de la circulation des utilisateurs, nous disposons de nouvelles dimensions d'indexation perçues par le cache. A partir de ces dimensions, nous proposons une nouvelle politique de gestion du cache et une nouvelle technique de prefetching. De plus, cette sémantique induite révèle des informations sur les documents eux-mêmes et nous signalons déjà quelques possibilités d'exploiter cet enrichissement. Nous reviendrons sur la mise en place de services mettant ces informations à disposition des utilisateurs dans le chapitre 7.

1.1 L'espace Web

Le World Wide Web se présente comme un espace de documents et de navigation. Ainsi, « The system we need is like a diagram of circles and arrows, where circles and arrows can stand for anything »⁶ comme le décrit Tim Berners-Lee dans [6], donne autant d'importance aux noeuds (documents) qu'aux liens qui les connectent : « Circles and arrows leave one free to describe the interrelationships between things »⁷. Notons qu'à l'origine Tim Berners-Lee souhaitait donner une valeur sémantique aux liens en les typant (A is part of B, A Made B, A uses B, A refers B) comme décrit dans [39]. C'est d'ailleurs le même style de liens typés que l'on retrouve dans les modélisations UML, Entité/Association ou encore DAML pour les ontologies. Cette typologie des liens a ensuite été abandonnée avec la normalisation des URL pour la définition des liens qui relient les documents, d'où une perte sémantique dans l'élaboration du Web.

1.1.1 Le Webgraph

De nombreuses analyses ont été effectuées sur le Web en tant que graphe. Nous prendrons [63] comme référence, avec les définitions : « The directed graph induced by the hyperlinks between web pages; we refer to this as the web graph »⁸ ainsi que : « nodes represent static html pages and hyperlinks represent directed edges »⁹. L'article s'appuie sur le fait que l'étude du "webgraph" a déjà permis de nombreuses améliorations sur le plan des moteurs de recherche et de classification des documents du web.

Nous constatons que dans les différents systèmes d'indexation proposés pour les caching proxys, toute cette dimension est ignorée. Ceci s'explique probablement par deux raisons : d'une part, les techniques de cache classiques sur lesquelles se sont d'abord appuyés les concepteurs en mettant en place des algorithmes tels que LRU ou LFU ne prennent pas en compte la disposition spatiale des éléments cachés et encore moins la possible valeur sémantique de ces derniers ; d'autre part, dans un souci de performance, le système de stockage est trivial et ne permet pas de stocker des relations entre les objets.

Or, si la disposition spatiale des éléments a une valeur pour les moteurs de recherche et d'autres applications similaires, il est évident qu'elle a aussi une influence sur le comportement des utilisateurs qui naviguent sur le Web, et donc sur l'usage qu'ils font du proxy. La perte en performances que pourrait entraîner l'intégration d'un système de stockage plus complexe en regard du gain sémantique nous paraît négligeable. Nous disposons de nos jours de SGBD, qui rivalise avec les performances des filesystems. Rien n'empêche, comme nous allons le montrer plus loin, de décomposer le schéma en sous-parties qui peuvent être traitées par des sous-systèmes indépendants, et donc de ne pas ajouter de latence dans le traitement des requêtes, même si l'on alourdit globalement la charge de traitement du système.

Enfin, le fait de pouvoir combiner les liens entre les documents et l'usage des utilisateurs, à savoir les chemins parcourus par ces derniers, nous semble d'une valeur sémantique intrinsèque impossible à ignorer. Comme nous allons le démontrer plus loin, nous induisons ainsi la notion de proximité entre les documents.

C'est un concept de localisation qui nous paraît extrêmement important. En effet, l'esprit humain a tendance à repérer la localisation des objets par les connexions qui les lient. On

⁶ Le système dont nous avons besoin est comme un diagramme avec des cercles et des flèches où les cercles et les flèches représentent n'importe quoi.

⁷ Les cercles et les flèches nous permettent de décrire librement les relations entre les choses.

⁸ Nous appelons webgraph le graphe orienté induit par les hyper-liens entre les pages du web.

⁹ Les noeuds représentent des pages HTML statiques et les hyper-liens représentent les arrêtes orientées.

se rappelle en effet autant, et parfois même mieux, du chemin qui lie deux points que de leur position absolue, et l'on est ainsi parfois surpris de découvrir sur une carte la position de lieux très familiers (par exemple dans la ville où l'on habite).

Définition 6: Le webgraph est le graphe composé d'arêtes orientées représentant les liens hyperlink reliant les documents et de nœuds représentant les ressources du web.

1.1.2 *La position des documents sur la toile*

Le deuxième élément de spatialité auquel nous nous sommes intéressés pour nos travaux est la localisation par les URL. Si les hyperlinks entre les documents représentent la position relative des documents entre eux, les URL représentent la position absolue des documents sur le Web. Dans [64], les URI ne sont définis que comme des identifiants et les URL comme l'expression de ces identifiants permettant de localiser sur le Web la ressource identifiée. Plus précisément, les URL sont définis comme : "The term Uniform Resource Locator (URL) refers to the subset of URI that identify resources via a representation of their primary access mechanism (e.g., their network "location"), rather than identifying the resource by name or by some other attribute(s) of that resource". Ainsi, les différents éléments qui constituent l'URL n'ont qu'une sémantique fonctionnelle permettant d'accéder à la ressource et non en relation avec la valeur sémantique de la ressource elle-même.

Cependant, des travaux comme [63, 65] prouvent que l'on peut tout de même trouver une valeur sémantique intrinsèque dans les URL. Si l'on pense aux processus de conception des sites Web : groupement des documents similaires dans des répertoires, site dédié à un domaine, domaine regroupé autour des activités d'une organisation, etc. Tout concorde pour appuyer le fait que des éléments d'un URL, comme l'adresse du serveur (donc le domaine où se trouve ce serveur) ou le chemin d'accès à la ressource, sont autant d'éléments qui constituent une sorte de raffinement sémantique qui nous rapproche de la ressource en question. Ainsi, on retrouve dans [64] la phrase suivante qui confirme la valeur intrinsèque des URL: "A URI often needs to be remembered by people, and it is easier for people to remember a URI when it consists of meaningful components"¹⁰. Prenons par exemple l'URL <http://lbd.epfl.ch/f/staff/stefano/index.html> nous pouvons en déduire que Stefano fait partie de l'équipe du LBD, qui est une unité de l'EPFL et que ce document se trouve dans la partie française du site.

Bien entendu, un certain nombre de cas viennent infirmer cette hypothèse. Citons comme exemples:

- Les pages personnelles qui présentent parfois des sujets en totale disjonction avec le site où elles se trouvent.
- Les sites fondés intégralement sur un SGBD avec des documents qui sont repérés par des identifiants automatiques.

Enfin, la profondeur des sites varie énormément.

On peut dire que dans le 90% des cas, les URL ont la signification attendue. Dans les 10% restants, on trouve des sous-sites qui sont en contradiction avec la thématique englobante et des sites où l'on ne peut pas distinguer de regroupement des documents (par exemple, documents identifiés par des identifiants énumérés).

¹⁰ Un URI doit souvent pouvoir être mémorisé par les gens, et c'est plus facile pour eux de le mémoriser lorsqu'il contient des composants qui présentent une signification.

1.2 Indexation du cache en fonction de la localisation des objets

Nous aborderons plus loin des propositions de stockage qui nous permettent de conserver différentes informations dans le cache. Nous faisons ici l'hypothèse que nous avons à notre disposition le moyen de conserver les dimensions et les paramètres que nous venons de décrire ci-dessus. Nous présentons ici une méthodologie qui nous permet d'extraire et d'exploiter ces dimensions pour la gestion du cache (nous verrons également plus loin d'autres applications pouvant exploiter ces nouvelles dimensions, c.f. prefetching).

1.2.1 Le Webgraph

Selon [12], lorsqu'un utilisateur suit un lien dans un navigateur, la requête http envoyée contient dans son en-tête le champ `referer` qui indique le document référençant le document référencé qui fait l'objet de la requête. Ce champ est absent lorsque la requête n'a pas pour source une ressource dotée d'un URI (par exemple dans le cas d'une adresse entrée au clavier). Il est donc facile pour un proxy d'extraire systématiquement cette information quand elle est présente et donc de pouvoir stocker non seulement les liens qui connectent les documents entre eux, mais aussi la fréquentation de ces "chemins". Nous avons donc à notre disposition un graphe orienté (qui se rapproche de l'idée originale présentée en [6]) avec des nœuds pour les documents et des arêtes qui les relient.

Nous définissons alors pour le webgraph:

$G_{web} = (V, E)$ avec $E = \{e_1, e_2, \dots, e_n\}$ un ensemble de nœuds qui représentent des ressources disponibles sur le Web et $V = \{v_{11}, v_{12}, \dots, v_{ij}, \dots, v_{mn}\}$ un ensemble d'arêtes qui représentent les liens entre ces ressources avec v_{ij} l'arête orientée qui relie le nœud e_i à e_j

Et la fonction:

$c: E \rightarrow \mathbb{R}^+$ qui associe à chaque arête un poids :

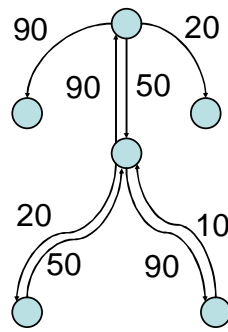
$$\left\{ \begin{array}{l} c(v_{kl}) = \infty \text{ pour une arête qui n'existe pas, c-à-d. qui n'a jamais été parcourue} \\ c(v_{kl}) = C, C \gg 0 \text{ pour un chemin à sa création, c-à-d. la première fois qu'un} \\ \text{utilisateur le parcourt avec } C \text{ une valeur constante.} \\ c(v_{kl})_{t+1} = r \cdot c(v_{kl})_t \text{ chaque fois que l'arête est parcourue une nouvelle fois,} \\ \text{avec } 0 < r < 1, \text{ facteur de réduction constant.} \end{array} \right.$$

Cette fonction associe à chaque arête une longueur constante à leur création, et surtout réduit les arêtes lorsqu'elles sont parcourues, ce qui a pour effet de rapprocher les nœuds d'autant.

Afin d'éviter une implosion du graphe avec le temps, nous imposons une longueur moyenne constante pour les arêtes. Nous appliquons alors (régulièrement):

$$c(v_{kl}) = x \cdot C(v_{kl}), \text{ avec } x = \frac{A \cdot m \cdot n}{\sum_{i=1, j=1}^{m, n} c(v_{ij})}$$

Cette fonction peut être appliquée soit à intervalle régulier, soit lorsque le poids d'une arête atteint un seuil minimum. Dans tous les cas, au fil du temps, l'évolution du graphe se traduit par l'éloignement et respectivement le rapprochement relatif des nœuds.



C = 100

Figure 14 : WebGraph

1.2.2 Topologie basée sur les URL

Comme nous l'avons présenté précédemment, les URL peuvent être utilisés comme un système de coordonnées des documents. Nous allons présenter ici un système de règles qui nous permet, à partir de cette composante de localité, d'extraire une dimension qui traduit la fréquentation des utilisateurs.

Hormis les informations liées au protocole (scheme) et à l'identification (username, password) [64], on trouve en le décomposant les lexèmes suivants : l'adresse du serveur et le chemin d'accès à la ressource. Nous réordonnons ces lexèmes: l'adresse dans l'ordre inverse, et le chemin (filepath) dans l'ordre direct. Nous avons ainsi des lexèmes qui sont des composantes de la clé d'accès dans l'ordre du plus général au plus spécifique.

Définition 7: La collection lexicale de l'URL est l'ensemble ordonné du plus général au plus spécifique des lexèmes qui ne sont ni l'élément protocole ni des éléments de l'authentification.

Exemple: à `http://lbd.epfl.ch/f/staff/stefano/index.html` correspond:

$$S = \{ "ch", "epfl", "lbd", "f", "staff", "stefano", "index.html" \}$$

Nous construisons alors un arbre lexicologique, en ajoutant à l'arbre pour chaque URL correspondant à une ressource accédée, le set S correspondant à cet URL.

Définition 8: L'arbre lexicologique est le graphe formé par le rattachement de toutes les collections lexicales à une même racine.

Soit : $G_{URL} = (V, E)$ avec $E = \{s_0, s_1, s_2, \dots, s_n\}$ un ensemble de nœuds correspondant aux lexèmes des URL visités, sauf s_0 la racine qui n'a aucune sémantique et $V = \{v_{01}, v_{02}, \dots, v_{ij}, \dots, v_{mn}\}$ l'ensemble des arêtes orientées qui relient ces nœuds, en partant de la racine s_0 vers les feuilles (les ressources elles-mêmes).

Nous définissons:

$K = \{k_1, k_2, \dots, k_n\}$ un ensemble de constantes numériques.

D tel que $D \gg \sum_{i=1}^n k_i$, une constante générale pour tout l'arbre.

Pour un URL donné, nous avons $S_{URL} = \{s_1, s_2, \dots, s_n\}$ une collection lexicale et $V_{URL} = \{v_1, \dots, v_m\}$ le chemin dans le graphe qui relie la racine à la feuille représentant la ressource identifiée, nous définissons alors une fonction :

$c: E \rightarrow \mathbb{R}^+$ qui associe à chaque nouvelle arête un poids,

$c(v_i) = k_i$ pour $i < m$

$$c(v_m) = D - \sum_{j=1}^{j < m} c(v_j) \quad (1)$$

$$\text{ou } c(v_m) = D - \sum_{j=1}^{j < m} k_j \quad (2)$$

par contre si l'arête v_j existe déjà, nous appliquons alors :

$c(v_j) = r \cdot c(v_j)$ avec $0 < r < 1$ un facteur de réduction constant.

Nous avons donc un graphe dont toutes les feuilles sont disposées à une distance D de la racine, constante à la création, et qui se rapprochent de la racine au fur et à mesure qu'elles sont fréquentées.

Afin d'éviter une implosion sur la racine de l'arbre avec le temps (toutes les feuilles à une distance infinitésimale de la racine), nous appliquons (régulièrement) :

Pour tout $v_{ij} \in V = \{v_{01}, v_{02}, \dots, v_{ij}, \dots, v_{nm}\}$:

$c(v_{ij}) = c(v_{ij}) + M$ avec M une valeur constante arbitraire.

Ce que nous obtenons est un arbre lexicologique des URL traversés, avec les arêtes des éléments lexicaux apparaissant souvent, qui se raccourcissent, alors que celles représentant des éléments plus rares se rallongent.

Nous obtenons également un effet d'activation du voisinage. En effet, un URL qui partage des lexèmes avec d'autres URL bénéficie aussi de l'activation de ces derniers qui induisent le raccourcissement d'arêtes qu'ils partagent et le rapprochent de la racine.

Nous avons le choix entre deux possibilités : soit les nouvelles ressources bénéficient directement de l'activité du voisinage (2), ou alors toutes les nouvelles ressources sont à la création placées à distance égale de la racine (1). Dans les deux cas, on observe une évolution du graphe similaire à ce que l'on observe dans les réseaux auto-organisés type Kohonen [66].

Définition 9: L'altitude d'une ressource dans l'arbre lexicologique est l'élévation en direction de la racine en fonction de l'évolution du graphe que subit la feuille représentant cette ressource.

Cette altitude peut être représentée par :

$$A_{URL} = D - \sum_{j=1}^n c(v_j)$$

Elle prend des valeurs négatives dans le cas de ressources peu actives.

1.2.3 Renforcement de l'activation de voisinage

Comme nous l'avons vu, le webgraph, représente une topologie qui décrit les distances relatives entre les ressources du Web en fonction des liens qui les connectent et de la fréquentation des utilisateurs qui suivent ces liens, alors que l'arbre lexicologique traduit une élévation des ressources les plus actives avec, de plus, une influence de voisinage, avec un voisinage défini par la position absolue basée sur les URL des ressources et les lexèmes communs.

Nous souhaitons ici répercuter cette notion d'activation du voisinage sur le webgraph en activant les liens qui connectent les nœuds qui se trouvent à une certaine distance, en s'inspirant de ce qui est décrit dans [67].

Selon [68], la fonction qui représente le mieux la notion d'activation de voisinage, comme décrite dans les réseaux de Kohonen, est la fonction dite du chapeau mexicain, également utilisée dans les techniques de filtrage/transformation par wavelet dans des applications topologiques [69]:

$$f(x) = \frac{1}{\sqrt{2\pi}} (1-x^2) e^{-\frac{x^2}{2}}$$

A chaque activation d'un nœud dans le graphe lexicologique, nous appliquons au nœud correspondant à la même ressource dans le webgraph:

$$d_{kl} = \sum_{p=k, q=i}^l c(v_{p,q}) \text{ la distance qui sépare le nœud } e_k \text{ du nœud } e_l$$

$MIN(d_{kl})$ le chemin le plus court qui sépare le nœud e_k du nœud e_l

$N_k = \{n_1, n_2, \dots, n_n\} | \forall n_k \in V, MIN(d_{kl}) < D$ l'ensemble des nœuds qui se trouvent dans le voisinage de e_k étant donnée une distance D , et V_k l'ensemble des arêtes qui relient ces nœuds.

Pour chaque arête faisant partie de N_k on applique alors:

$$c(v_n)_{t+1} = c(v_n)_t \cdot (1 - (f(\alpha \cdot d_{kl}) \cdot \beta)) \text{ avec } 0 < \beta < 1, 0 < \alpha$$

1.3 Applications à la gestion du cache

En résumé, nos deux graphes se construisent et évoluent en fonction de la position des documents sur le Web, la position relative des documents, que ce soit par les liens qui les connectent ou par les portions d'URL qu'ils partagent, et surtout dans le temps, en fonction de la fréquentation des utilisateurs. Ce dernier point nous permet de proposer une gestion du cache similaire aux politiques déjà proposées, mais qui, de plus, tient compte de l'espace Web que construisent les ressources dispersées sur la toile.

Au lieu de prendre comme paramètre de décision d'éviction des documents du cache le nombre de fois qu'un document a été demandé, nous prendrons l'altitude comme définie dans la fonction ci-dessus. Moins celle-ci est élevée, plus le document devient candidat à l'éviction.

Comme cette altitude n'est plus seulement fonction du nombre d'accès, mais également de l'activité dans le voisinage, notre politique d'éviction prend ainsi en compte la topologie de l'espace Web comme modélisée dans les différentes fonctions qui précèdent.

2 Le prefetching

Comme nous l'avons présenté dans l'état de l'art, le prefetching consiste à anticiper l'usage futur des clients et à initier des (télé-)chargements en conséquence. Les

algorithmes existants se basent sur des statistiques portant sur un voisinage immédiat des ressources.

Le prefetching consiste à anticiper les requêtes qui vont être effectuées par les clients et à cacher les documents avant que l'utilisateur n'ait effectué la requête. Cette technique ne permet pas de faire des économies de consommation sur le réseau, mais d'augmenter la vitesse apparente pour les utilisateurs. Le prefetching permet d'effectuer deux actions : placer le document en mémoire s'il se trouve dans l'espace de stockage disque du cache ou effectuer la requête au serveur d'origine s'il ne s'y trouve pas.

Nous allons ici présenter une solution originale pour exploiter l'indexation que nous proposons pour mettre en place des fonctions de prefetching en nous basant sur une modélisation fondée sur des chaînes de Markov.

2.1 Les chaînes de Markov

En reprenant le début d'analyse proposé dans [70], qui établit que la popularité du serveur et que le chemin d'accès d'un document sont des paramètres à prendre en compte pour établir la probabilité qu'on accède à ce document à nouveau, il apparaît intéressant de reprendre le webgraph que nous avons établi plus haut et qui modélise justement les éléments en question.

Des modélisations de la navigation en utilisant un modèle de Markov ont déjà été proposées, mais avec certaines limites (voir problèmes cités au chapitre 2).

[71] Soit processus stochastique constitué d'un nombre d'états finis et discrets, on définit alors pour la variable aléatoire X :

$$p_{ij}(n) = p(X_{n+1} = j | X_n = i)$$

la probabilité que la variable X passe de l'état i à l'état j au cours de l'intervalle temporel $[n, n+1]$.

On peut alors définir:

$(p_{ij}) = P$ la matrice des probabilités de transition ou matrice de transition.

On peut modéliser les états du processus par des points et les probabilités de transition par des arcs reliant ces points (

Figure 15).

Définition 10: Un état d'une chaîne est dit absorbant si une fois que le système est entré dans cet état, il ne peut plus le quitter.

Dans l'exemple donné, les états 2 et 4 sont dits absorbants.

Notons au passage que l'on a clairement la somme des probabilités sur une ligne de la matrice égale à 1.

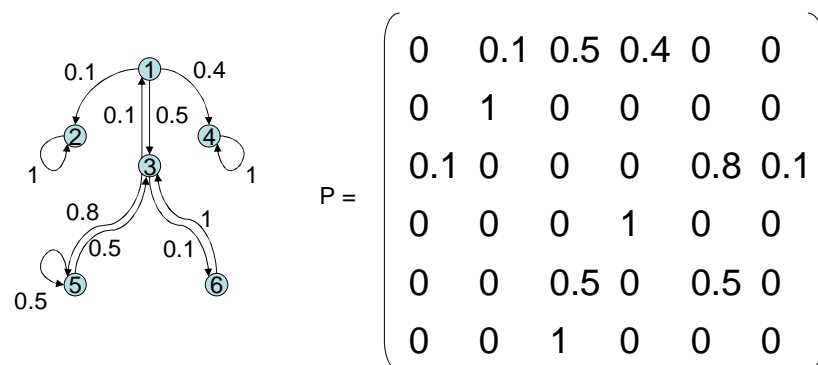


Figure 15 : Chaîne de Markov, graphe et matrice de transition

On peut alors introduire un état initial:

$$\pi(0) = (1 \ 0 \ 0 \ 0 \ 0 \ 0)$$

Et déterminer les probabilités d'état pour l'instant n :

$$\pi(n) = \pi(0)P^n$$

Dans notre exemple, à l'instant 2, les probabilités seront donc:

$$\pi(2) = \pi(0)P^2 = (0.05 \ 0.10 \ 0 \ 0.40 \ 0.40 \ 0.05)$$

Et l'instant 3:

$$\pi(3) = \pi(0)P^3 = (0 \ 0.105 \ 0.275 \ 0.420 \ 0.200 \ 0)$$

On peut donc, à partir d'un graphe d'états, calculer à différents instants futurs discrets, la probabilité des différents états du système.

2.2 Adaptation au Webgraph

On voit une similitude entre le graphe de l'exemple et le Webgraph (Figure 14) à ceci près que le Webgraph ne nous donne pas des probabilités mais des statistiques d'utilisation avec un poids inversement proportionnel à la fréquentation et nous remplaçons les états par des ressources Web. De plus, contrairement aux systèmes modélisables avec des chaînes de Markov, nous supposons un graphe avec un nombre d'états quasi-infini. Pour modéliser les probabilités de cheminements possibles à partir d'une ressource, nous définissons alors un degré de profondeur à partir de la ressource courante pour délimiter un graphe fini. Ce degré de profondeur définit le chemin maximum possible pour lequel nous appliquons notre algorithme. Pour chaque niveau de profondeur, nous avons donc un ensemble de nœuds:

$S_0 = \{e_0\}$, avec e_0 la ressource initiale

$S_1 = \{e_{11}, e_{12}, \dots, e_{1n}\}$ les ressources accessibles au degré 1, donc en parcourant un seul lien

$S_m = \{e_{m1}, e_{m2}, \dots, e_{mm}\}$ les ressources accessibles au degré m

Il nous faut donc adapter les valeurs de notre Webgraph et les normaliser pour obtenir une matrice de transition. Nous proposons une relation linéaire de premier degré entre la probabilité de suivre une arête et la fréquentation enregistrée.

Donc pour l'arête orientée v_{ij} qui relie la ressource e_i à la ressource e_j :

$$p_{ij} = \frac{v_{ij}}{\sum_{k=0}^n v_{ik}}$$

De plus, il nous faut normaliser ces probabilités de façon à ce que la somme des probabilités au départ d'un état soit toujours égale à 1, même pour les états qui n'ont aucune arête sortante.

$$p_{ii} = 1 \text{ si } \sum_{k=0}^n v_{ik} = 0$$

Toutes les ressources avec $p_{ii} = 1$ sont des états absorbants. Ces derniers posent un problème qui empêche l'évaluation à long terme. En effet, ils représentent très certainement des objets inclus dans une ressource. Ils sont donc très probables, mais ne présentent aucun lien vers une autre ressource et induisent des états fortement stationnaires qui faussent complètement l'évaluation du chemin probable.

Pour résoudre ce problème, nous proposons à chaque itération de supprimer les ressources absorbantes du niveau de profondeur courant.

Nous avons alors:

$(p_{ij}) = P_1$, la matrice de transition pour la première itération

Ensuite nous définissons P_2, P_3, \dots les matrices de transition successives dont nous supprimons chaque fois les ressources absorbantes de degré correspondant.

Nous avons alors pour connaître la probabilité d'état à l'instant n :

$$\pi(n) = \pi(0) P_n^n$$

Notons enfin la propriété suivante:

Soit P' la matrice de dimension $n \times n$ des transitions sans la ressource courante, alors $\text{Trace}(P') = n$ si et seulement si toutes les ressources sont des ressources incluses.

Démonstration: $\text{Trace}(P') = \sum_i p_{ii}$ la trace de P est donc la somme des probabilités

stationnaires. Si tous les éléments sont stationnaires (ressources incluses), alors $p_{ii} = 1 \mid i$.

3 Conclusion

Nous avons donc élaboré une nouvelle indexation de nos objets dans l'espace cache qui nous permet de tenir compte de la disposition des documents sur le Web, aussi bien la disposition relative qu'absolue. Cette indexation nous donne la possibilité d'étendre la perception du proxy et de mettre en place une politique de cache qui prenne en compte ces nouvelles dimensions. De plus, les informations statistiques que nous avons ainsi collectées nous permettent de mettre en place un mécanisme de prefetching qui, contrairement aux systèmes proposés à ce jour, nous autorise une évaluation à plus long terme des ressources probables dans le cheminement d'un utilisateur. Pour ce dernier point, les techniques de prefetching actuelles permettent surtout de pré-charger les objets qui sont inclus dans la ressource en cours de chargement, or, pour ces dernières, l'anticipation sur les actions entreprises par l'agent client est faible. En effet, la plupart des navigateurs entreprend par exemple le chargement des images incluses dans un document HTML au fur et à mesure de leur apparition dans le document. Avec notre algorithme, nous sommes capables d'anticiper, sans tomber dans des états stationnaires erronés, des chemins de longueur importante. Dans le cas de vitesse réseau fortement dégradée, nous proposons alors une vitesse apparente très importante.

Par extension, cette technique de prefetching serait également possible au niveau du client, ce qui découragerait l'utilisation d'outils qui permettent de télécharger à l'avance tout un site Web. Ces outils sont de gros consommateurs de bande passante en induisant un nombre élevé de téléchargements inutiles.

Par contre, nous avons vu que pour pouvoir construire correctement notre index, nous avons besoin de la directive HTTP `referer` que certains clients ne mettent pas dans leur requête. L'alternative serait de disposer du concept de session utilisateur pour être capable de détecter correctement les enchaînements qui constituent le chemin parcouru par un utilisateur. Nous allons donc proposer au chapitre suivant un moyen de disposer de cette information dans les proxys.

SESSIONS PROXY

Proposition d'une persistance d'état entre le proxy et le client reposant sur le principe de session

1 Introduction

Dans les chapitres précédents, nous avons présenté la situation privilégiée des proxys en définissant notamment le concept de plésiocentrisme. Nous avons également étendu la perceptivité du proxy en y ajoutant quelques éléments sémantiques (chapitre 3). Nous nous sommes cependant arrêtés aux utilisations "classiques" du proxy : à savoir un élément transparent, au point même d'être invisible pour les utilisateurs, qui n'offre qu'une fonctionnalité de caching et une augmentation de la vitesse apparente de la navigation.

Nous l'avons par contre évoqué à la fin du chapitre précédent, il nous manque des mécanismes pour pouvoir identifier un utilisateur et collecter avec fiabilité les statistiques de parcours. C'est pourquoi nous introduisons le concept de session qui permet l'identification de l'utilisateur au travers d'un enchaînement de transactions.

Dans la suite de notre travail, nous allons proposer l'introduction de services plus évolués sur la base de l'architecture générale des proxys. Mais ces services nécessitant la possibilité d'interagir avec l'utilisateur, nous allons, dans ce chapitre, non seulement introduire la notion de session mais également son application aux proxys.

Le maintien de cette session devant s'intégrer au protocole HTTP, nous allons dans un premier temps examiner cette problématique dans le cadre de ce protocole. Nous étudierons les différentes solutions proposées à l'heure actuelle. Puis, avant de présenter la solution retenue, nous évaluerons l'adéquation entre ces différentes techniques et l'objectif de supporter le paradigme de la session dans les interactions entre le client et le proxy.

Suite à cet état de l'art technique, nous proposons un nouveau concept : les proxy-cookies. A l'instar du cookie, le proxy-cookie est un mécanisme qui permet de "marquer" les clients, et donc le support de sessions.

1.1 Définition

Le concept de session en informatique correspond à l'idée que nous nous faisons d'une conversation dans la vie courante. Ainsi, les deux partenaires commencent par une identification, échangent des salutations et construisent un discours basé sur des échanges d'informations. Une session correspond donc typiquement à un point (client) qui envoie une demande de connexion à un autre point (serveur), une fois que les deux points se sont mis d'accord sur le protocole et les paramètres de la connexion, les deux parties échangent des informations, des commandes, avec un contexte qui évolue (enrichissement d'un côté ou de l'autre, acquisition de nouveaux paramètres associés à cette session, etc.). La session commence avec l'établissement de la connexion et prend fin avec la terminaison de cette dernière.

Définition 11: Dans le domaine des télécommunications, une session est une série d'interactions entre deux points qui se situent dans l'intervalle d'une seule connexion.

Le concept de session est crucial lorsque l'on veut proposer une interaction car c'est l'environnement qui lie les différentes transactions entre elles.

La session permet de créer un contexte.

Ainsi, la session permet d'assurer la persistance de l'état au cours d'une succession d'interactions correspondant à un service proposé au client. Il est donc évident qu'avant de pouvoir proposer des services, il fallait proposer une solution permettant l'établissement d'une session. Etant donné que le même problème s'est posé pour le protocole http en général et que c'est celui qui est utilisé entre le client et le proxy, nous avons naturellement exploré les solutions proposées dans ce domaine.

2 HTTP et session

Le protocole HTTP est sans connexion au niveau de la couche application, ce qui signifie que la connexion ne reste pas établie entre les différentes requêtes qu'un client envoie à un même serveur. Du fait que la connexion est abandonnée après chaque interaction, il n'est pas possible, au niveau du serveur, d'y associer des données qui puissent être reprises à l'interaction suivante. Le protocole HTTP est donc sans état (stateless).

Comme on le voit sur le schéma (Figure 16), il n'y a pas de distinction entre les trois requêtes. La seule différenciation possible entre l'opération 1 et l'opération 2 pourrait résider dans l'adresse réseau (adresse IP) des deux clients. Mais cette adresse n'est pas un identifiant valide, même si le cas n'est pas fréquent, deux utilisateurs différents peuvent très bien instancier deux applications clientes sur une même machine et donc partager la même adresse. Plus délicat encore : la distinction entre l'opération 1 et l'opération 3. Rien ne garantit qu'il s'agisse du même utilisateur.

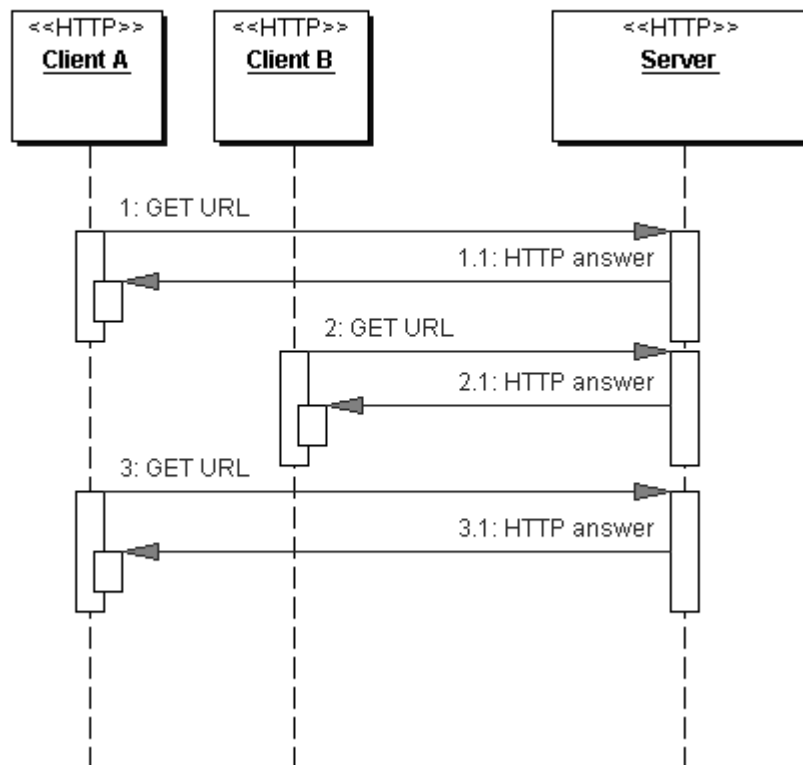


Figure 16 : HTTP stateless

Cette caractéristique n'a pas que des inconvénients : l'essence même du protocole http repose sur la dissémination des documents dans de nombreux serveurs. Par conséquent, le maintien d'une connexion n'a pas de sens et aboutirait rapidement à une prolifération ingérable des connexions en cours.

Par contre, il n'est pas possible de mettre en place des transactions qui dépassent une seule interaction, ce qui est une limitation sévère sitôt que l'on veut construire sur cette base des applications plus complexes que la simple distribution de ressources comme pour l'e-commerce, l'e-banking, etc.

Nous allons donc examiner dans ce qui suit les différentes solutions qui sont apparues pour pallier ce problème : l'authentification http, les champs cachés, la réécriture des URL et les cookies. Nous aborderons les techniques existantes dans l'ordre où elles sont apparues parce que cet ordre correspond à une évolution qui a abouti aux cookies. C'est la solution la plus complète et la plus répandue pour le maintien de sessions, même si la réécriture des URL reste encore très utilisée notamment pour supporter les clients qui n'acceptent pas les cookies.

2.1 *L'authentification http*

La première solution d'identification de l'utilisateur qui existait dès la première définition du protocole http par Tim Berners-Lee est l'authentification http. Elle se déroule comme suit : on définit du côté du serveur les documents qui nécessitent une authentification et on associe à cette définition une table de paires username/password. Lorsqu'une requête pour un tel document est envoyée au serveur, celui-ci vérifie que l'en-tête de la requête contient les informations d'authentification : une paire username/password et que celle-ci correspond bien à une paire valide. Si les informations d'authentification ne sont pas présentes ou ne sont pas valides, alors le serveur renvoie un code d'erreur '401 Unauthorized'. Le navigateur client propose alors à l'utilisateur de rentrer un username et un password, puis réitère la requête. Le processus se répète jusqu'à ce que l'authentification soit correcte, ou que l'utilisateur annule l'authentification du côté client et le browser affiche alors le message d'erreur.

Une fois que l'authentification a été acceptée, l'agent utilisateur renvoie alors à chaque transaction avec le même serveur dans l'en-tête de la requête la paire username/password. Une session peut donc s'établir en associant au username un contexte.

Cette technique a comme inconvénient majeur que l'information username/password est transmise en clair à chaque transaction et qu'elle ne permet pas (en tout cas pas de façon normalisée) la fin explicite de la session par l'annulation de l'information d'authentification. Enfin, autre inconvénient, une session ne peut exister que pour un utilisateur complètement authentifié. Il n'est pas possible de proposer des services à des personnes qui n'ont pas de compte (username/password) enregistré.

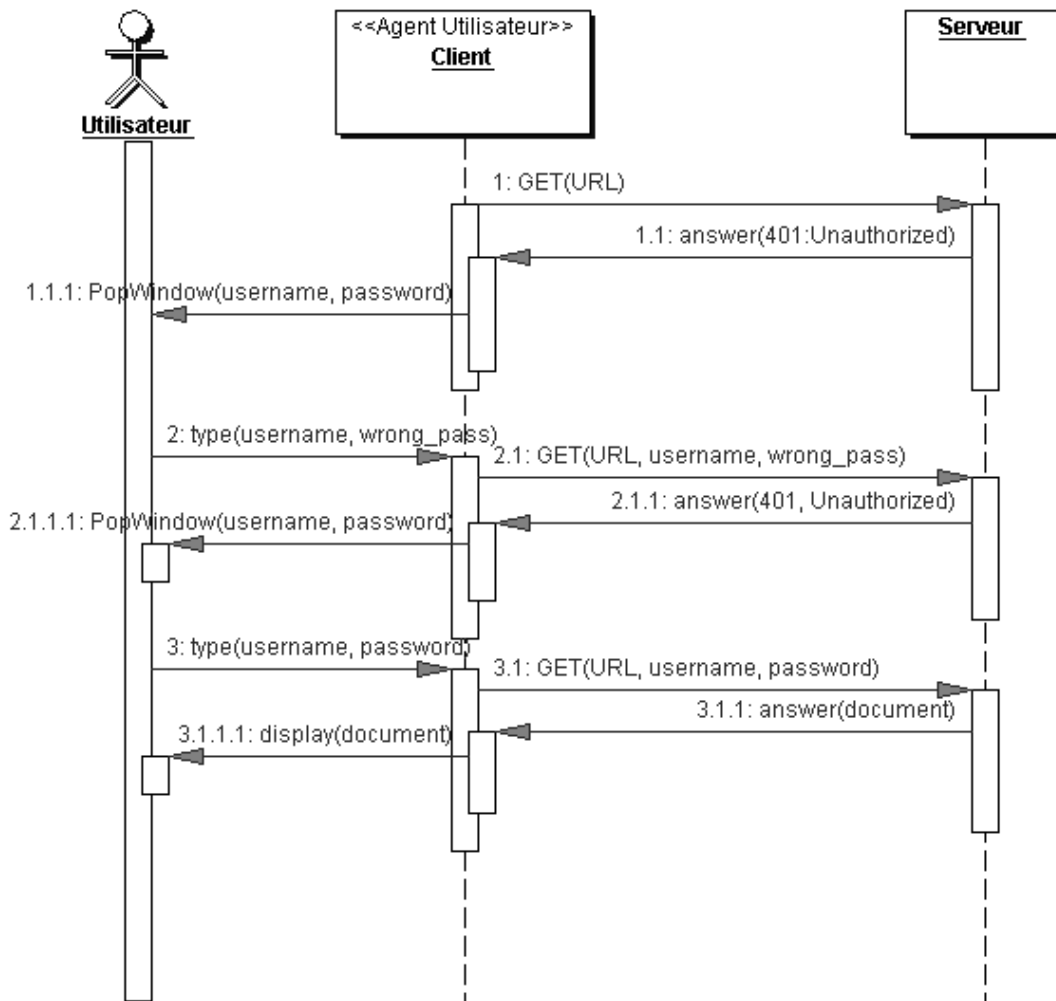


Figure 17 : Authentification HTTP

2.2 Les champs cachés

La première tentative d'établissement d'une session pour faire persister des informations en vue de l'élaboration d'un service sans imposer d'authentification utilisait la technique du champ caché.

Cette technique consiste à générer une clé de session que l'on stocke dans un champ caché d'un formulaire HTML. A chaque soumission du formulaire au serveur, celui-ci récupère la clé de session et renvoie un nouveau formulaire résultat qui contient de nouveau la clé de session dans le même champ caché.

Cette méthode a comme inconvénient quasi insurmontable de forcer à une navigation par formulaire uniquement. D'ailleurs, elle a été complètement abandonnée dès l'apparition des cookies.

2.3 La réécriture des URL

Cette solution repose sur la possibilité de faire passer des paramètres dans un URL. C'est en fait ce qui se passe dans les requêtes issues d'un formulaire utilisant la méthode GET. Ainsi, tous les URL contenus dans un document, et pointant sur d'autres documents faisant partie du même service, sont modifiés en y ajoutant à la fin un paramètre qui contient l'identifiant d'une session valide pour le serveur.



```
http://clue.veesion.com/app.php?SID=278
```

Figure 18 : URL avec paramètre

Cette méthode a comme avantage, par rapport aux cookies, que si l'on quitte un service, on perd inmanquablement l'information permettant de maintenir la session. Ainsi, un utilisateur peut librement clore une session quand il le souhaite.

Elle a cependant deux inconvénients. Le premier concerne la conception d'un service Web. On est en effet obligé de systématiquement réécrire tous les URL de tous les documents d'un même service. On se retrouve alors forcé de ne proposer que des documents générés dynamiquement qui échappent donc de manière généralisée au caching par les proxys. Cependant, la plupart des toolkits (API servlet, PHP, ASP, etc.) proposent des solutions quasi transparentes pour cette opération. Le deuxième inconvénient, beaucoup plus pernicieux, concerne la sphère privée. Des identifiants peuvent être cachés à l'utilisateur dans des liens où il ne soupçonne pas leur existence et permettre donc la transmission d'informations d'un site à un autre. Cette technique est largement utilisée dans les mails de Spam au format HTML : des liens avec des paramètres d'identification sont dissimulés dans le corps du document. Ainsi, dès que l'agent utilisateur "suit" un de ces liens, il renvoie cette information au serveur.

2.4 Les cookies

Peu de temps après les débuts du WWW, Netscape proposait une solution pour pallier ce problème et permettre de proposer des services (e-commerces ou autres) reposant sur le principe d'une session : les cookies.

2.4.1 Fonctionnement des cookies

Le principe de base des cookies [1] est d'inscrire une marque sur les clients. Les clients présentent ensuite cette marque à chaque requête. Le serveur a donc le moyen de générer des marques uniques, ou jetons d'identification, que le client présente. La séquence des événements avec l'utilisation de cookies (Figure 19) illustre ce fonctionnement. Aux étapes 1 et 2, nous voyons un client A qui reçoit un jeton d'identification et le présente ensuite à ce même serveur. Les étapes suivantes montrent comment les cookies permettent de distinguer deux clients.

Dans le cas d'une connexion, le cycle de vie d'une session commence avec l'établissement de la connexion et se termine avec la coupure de celle-ci. Avec les cookies, l'expiration d'une session ne peut pas être associée à un événement (dans l'exemple, entre les interactions 2 et 5, l'identification persiste), un intervalle temporel doit être explicité. C'est pourquoi les cookies disposent d'un attribut qui limite la validité du cookie.

Définition 12: *La validité d'un cookie est sa limite de validité dans le temps.*

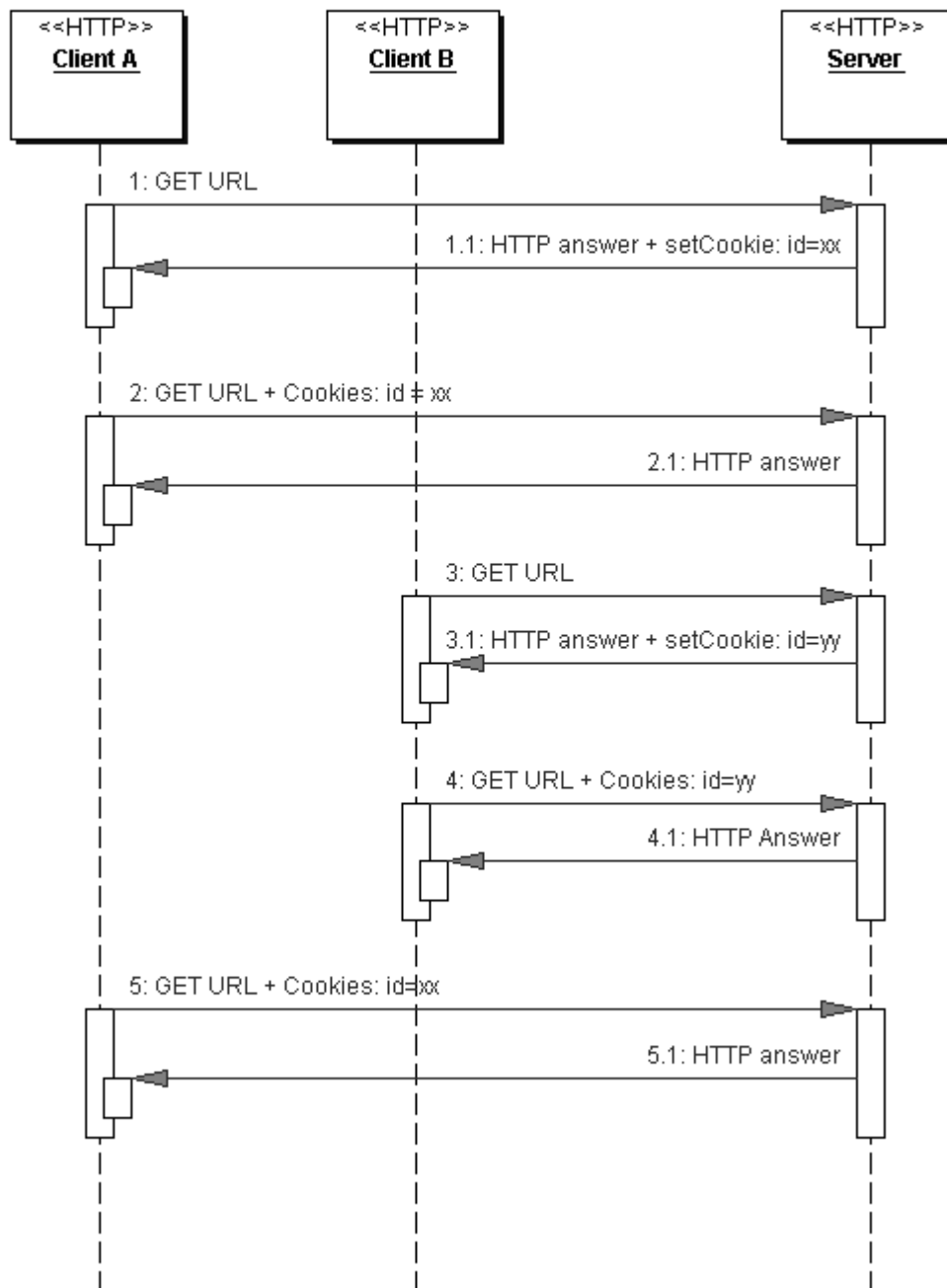


Figure 19 : les cookies: scénario

Comme autres paramètres, nous trouvons également parmi les plus importants : le domaine (exemple: epfl.ch, yahoo.com) pour lequel le cookie est valide; le path (chemin) qui limite éventuellement la validité du cookie à un sous-ensemble d'URL du même domaine.

Définition 13: La portée d'un cookie correspond à l'ensemble des URL pour lesquels il sera validé. Cette portée dépend du domaine et du chemin.

Dans les développements précédents, nous avons montré un exemple où seul un identifiant de session était stocké dans des cookies, ce qui est le plus courant. Bien entendu, d'autres informations peuvent être stockées dans les cookies, mais cette pratique n'est pas recommandée. Il faut garder à l'esprit que les cookies augmentent la taille des en-têtes HTTP à chaque requête.

On peut définir différents niveaux de session. Par exemple, une session sécurisée où l'on considère l'utilisateur comme complètement authentifié, avec une validité courte (généralement 15 min.) et une session longue pouvant s'étendre sur des mois où l'on tente de reconnaître l'utilisateur mais on ne le considère pas comme formellement authentifié¹¹.

2.4.2 *Limitations, problèmes*

Si les cookies répondent à la problématique de faire persister une session au-delà de la durée de vie d'une connexion, ils rencontrent cependant une assez forte résistance au niveau des utilisateurs. Il y a deux raisons à cela, toutes deux sont liées à la sécurité et plus particulièrement au respect de la sphère privée de l'utilisateur. Comme la session s'étend au-delà de la persistance de la connexion, elle échappe au contrôle de l'utilisateur qui ne peut plus « fermer » la session en coupant la connexion. Ainsi, les cookies peuvent être affublés d'une durée de vie quasi infinie et, longtemps après, un site sera encore capable de vous identifier et donc de suivre vos actions ainsi que d'analyser votre comportement à votre insu. D'autre part, beaucoup d'utilisateurs craignent que leur identification au travers d'une session sur un site ne soit ensuite disponible pour les traquer sur un ensemble de services Internet.

Cette résistance n'est que partiellement fondée et doit certainement sa force à un battage médiatique exagéré lors de l'apparition des cookies vers 1995. En effet, presque tous les agents utilisateurs limitent par défaut l'activation des cookies au serveur d'émission. Et plus amusant, pour tous ceux qui désactivent les cookies sur leur navigateur, il existe des solutions alternatives qui elles par contre ne sont pas soumises à ce genre de limitation ! En revanche, il est vrai que l'utilisateur n'a pas de contrôle direct sur la durée de vie de cookies et peut donc rester identifié sur un site malgré lui et à son insu.

2.5 *Conclusion*

Comme nous sommes à la recherche d'une solution qui permette l'établissement d'une session entre le client et le proxy, nous exposerons également quelques considérations sur l'adéquation entre les techniques présentées et notre objectif.

Si l'on reprend les différentes formes de maintien de session que nous avons présentées, nous avons écarté les autres solutions que les cookies pour les raisons suivantes. On voit immédiatement que la méthode des champs cachés n'est pas applicable puisqu'elle se limite aux formulaires. Pour la réécriture des URL, il aurait bien entendu été possible de réécrire à la volée tous les URL des documents passant à travers le proxy. Cependant, il aurait été difficile de réserver un domaine de variables pour notre application afin de s'assurer qu'aucun service n'utiliserait jamais le même lexème comme nom de variable pour le paramètre de session. D'autre part, la surcharge en terme de traitement au niveau du proxy, alors que les performances sont justement un facteur clé, aurait été énorme. L'utilisation d'une version altérée de l'authentification http était envisageable. Elle aurait cependant procédé des mêmes inconvénients que l'originale (nécessité d'un compte utilisateur pour chacun) et enfin elle aurait demandé à l'utilisateur une interaction (fenêtre d'authentification) invasive et propre à semer la confusion. Enfin, d'autres solutions auraient pu être proposées comme l'insertion dans tous les documents HTML d'un

¹¹ Il pourrait s'agir d'une autre personne utilisant le même agent utilisateur.

élément d'en-tête sous forme de tags <meta>. Mais nous avons écarté cette dernière solution parce que limitée à une certaine catégorie de documents (HTML !) et peu généralisable. Enfin, elle aurait posé de sérieux problèmes au niveau du caching, puisque les ressources deviennent variables (chaque utilisateur obtient une ressource modifiée différente !).

3 Proposition d'un système de maintien de session pour les proxys

Dans la définition d'un mécanisme de maintien de la session entre le client et le proxy, nous nous sommes attachés à garantir l'utilisation transparente du proxy pour le client et d'appliquer un minimum d'altérations aux protocoles et aux normes existantes.

Au vu des considérations qui précèdent, nous avons choisi de proposer une solution basée sur les cookies. La première raison de ce choix est d'éviter d'avoir à intervenir dans les documents en transit (gain en temps, problème de cache) et de n'intervenir qu'au niveau du protocole HTTP, des en-têtes des requêtes et des réponses. De plus, les cookies standard représentent une technique déjà éprouvée et largement répandue. En choisissant ce paradigme pour notre solution, nous limitons les risques évoqués dans la partie précédente. Même si notre intention de ne pas avoir à apporter de modifications aux standards existants n'est pas complètement respectée, nous minimisons cependant l'impact sur les outils utilisés aussi bien au niveau du client que du proxy. Enfin, une implémentation sera facile à mettre en place en se basant sur les fonctions déjà implémentées pour les cookies.

Nous allons d'abord présenter le fonctionnement général des proxy-cookies avec les différents mécanismes qui entrent en jeu. Nous illustrerons ce fonctionnement avec des scénarios d'utilisation. Nous aborderons ensuite les problèmes qui y sont liés : l'intégration dans des arborescences de proxys et la sécurité. Nous terminerons par une définition plus formelle des proxy-cookies.

3.1 Les proxy-cookies

Les proxy-cookies sont donc une variante des cookies. Pour cette raison nous allons les présenter en nous référant au document de spécification des cookies [1].

Le fonctionnement des proxy-cookies est similaire au fonctionnement des proxys standard, mais il s'agit dans ce cas pour le proxy d'établir sur le client une marque d'identification. Le scénario (Figure 20) est donc très proche de celui appliqué pour les cookies.

Les principales différences résident dans la gestion au niveau du client. Celui-ci ne sait pas forcément à quel(s) proxy(s) il s'adresse, soit parce que la requête est relayée de proxy en proxy, soit parce qu'une redirection automatique est effectuée au niveau du routeur (cas fréquent chez les ISP: tout le trafic sur le port 80 est redirigé sur un proxy). Il ne peut donc pas y avoir de règles limitant la portée du proxy-cookie en fonction du domaine ou du path (chemin), puisque l'URL ne détermine pas complètement le ou les proxys adressés.

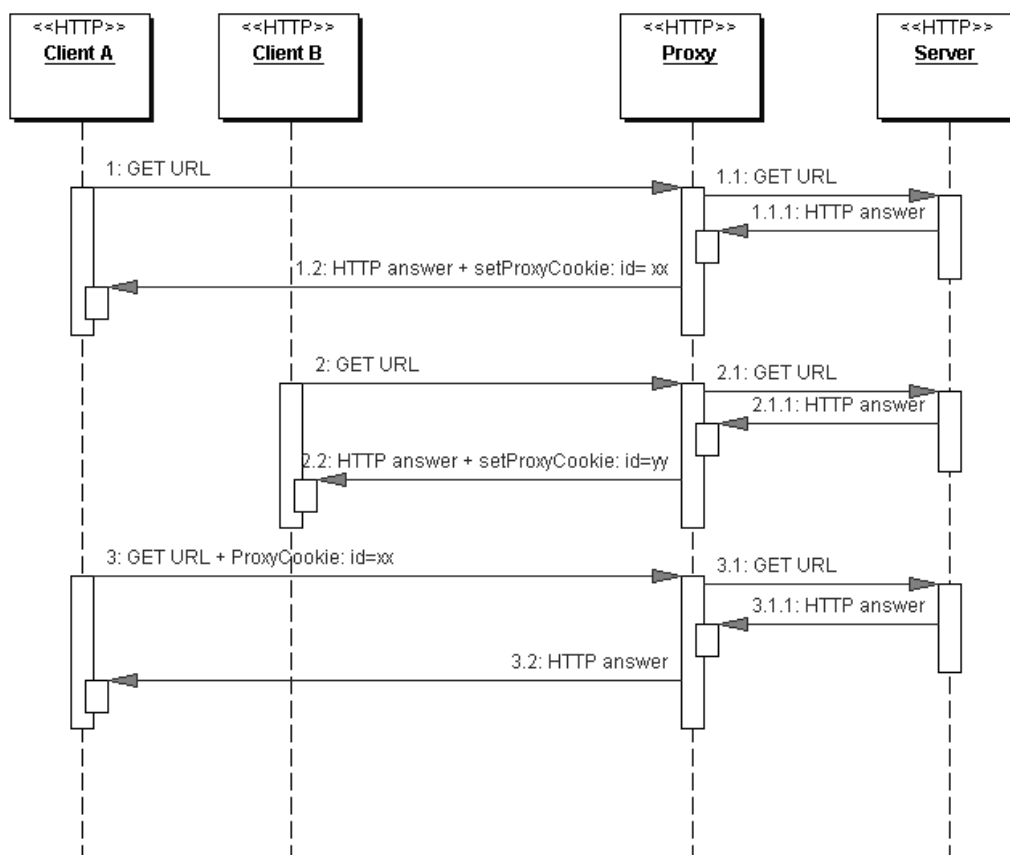


Figure 20 : Proxy-cookies: scénario

Notons au passage que les proxy-cookies sont parfaitement orthogonaux aux cookies standard. L'utilisation des proxy-cookies ne change rien aux interactions possibles avec le serveur pour les cookies.

3.1.1 Arborescence de proxys

Comme nous l'avons déjà évoqué au début de ce travail, les proxys sont souvent disposés en arborescence. Une requête adressée à un proxy est ainsi relayée à un proxy parent, et ainsi de suite si nécessaire, jusqu'à ce qu'elle soit satisfaite par un hit ou qu'elle soit relayée jusqu'au serveur d'origine. Cette structure hiérarchique est complètement invisible pour le client.

Afin d'éviter des recouvrements lexicaux pour la dénomination de proxy-cookies émis par des proxys différents, il nous faut ajouter une identification de l'émetteur dans les attributs du proxy-cookie. Ceci nous permettra par exemple de distinguer deux proxy-cookies émis par deux proxys différents, mais tous deux appelés "SID".

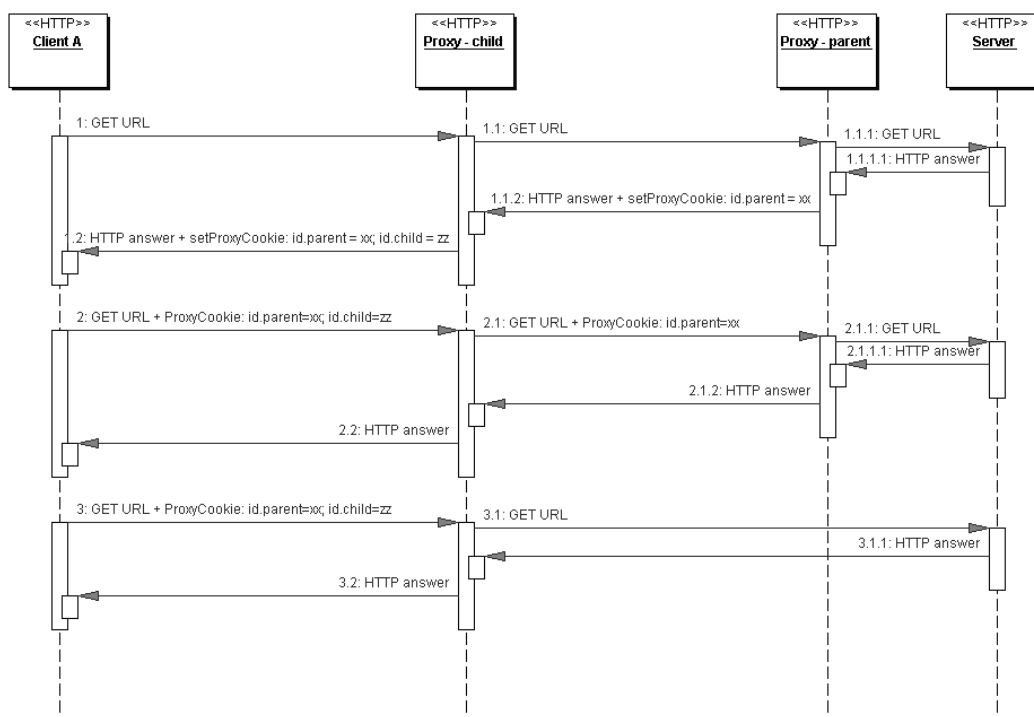


Figure 21 : Proxy-cookies: scénario avec hiérarchie de proxys

Notre paradigme supporte donc l'intégration à une échelle plus large, sans avoir à mettre en place un mécanisme de contrôle ou de supervision.

3.1.2 Sécurité

Il n'est pas acceptable que les proxy-cookies soient relayés au serveur d'origine. Comme on le voit dans les scénarios présentés ci-dessus (Figure 20, Figure 21), les proxy-cookies sont uniquement inclus dans les paramètres des requêtes adressées de client à proxy ou de proxy à proxy. En effet, cela permettrait de "tracer" un client dans sa navigation au-delà des limites d'un seul site. Si la spécification RFC 2965 [1] précise clairement qu'un cookie ne peut pas avoir une portée plus grande qu'un domaine (à l'exclusion des domaines racines), on ne peut pas, au travers du proxy-cookie qui a une portée illimitée, relayer une identification aussi générale jusqu'aux serveurs d'origine des documents.

En revanche, il est clair que les proxy-cookies permettent aux proxys d'évaluer le chemin parcouru par la requête au travers des différents proxys en amont de la requête (côté client). Mais cette information était déjà disponible (sauf cas de configurations particulières visant à faire disparaître cette information) dans la directive HTTP X_FORWARDED_FOR qui permet d'indiquer les intermédiaires par lesquels une requête transite (voir ch. 2).

La limitation à certains numéros de port reste pertinente. Dans le cas de la redirection automatique (chez certains ISP comme évoqué auparavant), la redirection n'aura pas lieu pour certains URL impliquant des connexions sur des numéros de port qui ne sont pas redirigés. Dans ce cas, le proxy devra être configuré correctement et n'émettre que des proxy-cookies limités aux ports redirigés sous peine d'introduire des brèches dans la sécurité (voir plus loin).

3.1.3 Définition formelle

Nous définissons formellement les proxy-cookies à partir des définitions données dans [1] qui formalisent la gestion d'état dans le protocole HTTP (HTTP State Management Mechanism, RFC 2965). Cette gestion se fait au moyen des cookies, ce qui se traduit dans le protocole HTTP par l'introduction de deux directives¹².

En empruntant exactement la même démarche pour les directives set-cookie2 et cookie qui permettent de gérer les cookies, nous introduisons deux directives: respectivement set-proxy-cookie et proxy-cookie qui permettent de gérer les proxy-cookies et dont nous donnons la grammaire dans les encadrés.

```
set-proxy-cookie = "Set-Proxy-Cookie:" cookies
cookies          = 1#cookie
cookie          = NAME "=" VALUE *("; set-cookie-av)
NAME            = attr
VALUE          = value
set-cookie-av   = "Comment" "=" value
                | "CommentURL" "=" <"> http_URL <">
                | "Discard"
                | "Issuer" "=" hostaddress
                | "Max-Age" "=" value
                | "Port" [ "=" <"> portlist <"> ]
                | "Version" "=" 1*DIGIT
portlist        = 1#portnum
portnum         = 1*DIGIT
```

Définition 14: Directive set-proxy-cookie

Par rapport à ce qui est défini dans [1], il faut noter pour la directive set-proxy-cookie que nous avons supprimé les attributs domain, path et secure. Pour le domain et le path, comme nous l'avons dit précédemment, ils ne peuvent pas s'appliquer. Les proxy-cookies doivent être systématiquement envoyés par l'agent dès que la transaction passe au travers d'un proxy, quel que soit l'URL invoqué. Comme le domain et le path sont destinés dans le cadre des cookies standard à limiter l'envoi des cookies en fonction de l'URL, ces attributs devaient être supprimés.

L'attribut secure indique à l'agent qu'il ne doit renvoyer le cookie en question que si le niveau de sécurité de la transaction en cours est le même que celui de la transaction au cours de laquelle le cookie a été envoyé au client. Comme le niveau de sécurité est déterminé par le serveur d'origine, indépendamment du proxy, cette directive n'est pas applicable et pourrait aboutir sur la perte de proxy-cookies par le fait d'un changement de niveau de sécurité contrôlé par un serveur d'origine.

Par contre, nous avons ajouté l'attribut Issuer. Celui-ci permet d'identifier le proxy qui a émis le cookie. Il permet de distinguer des cookies émis par deux proxys différents mais ayant le même nom.

¹² Plus rigoureusement, trois directives, mais l'une sert uniquement au contrôle de version et nous ne rentrerons pas dans ces détails. Notons à ce propos qu'il existe globalement deux formes pour l'échange de cookies, la première ayant été définie par Netscape (voir aussi RFC 2109) et la seconde est celle décrite dans le document cité. Les deux versions coexistent à l'heure actuelle, mais selon toute logique, la dernière spécification donnée dans les RFC fait foi et nous n'avons tenu compte que de celle-ci dans notre travail, même si elle n'est pas encore très largement supportée. De toute façon, ces deux approches sont très similaires et pour les points qui les différencient, il n'y a aucun impact sur notre proposition.

```
Proxy-cookie      = "Proxy-Cookie:" cookie-version 1*((";" | ",")
cookie-value)
cookie-value      = NAME "=" VALUE [";" issuer] [";" port]
cookie-version    = "$Version" "=" value
NAME              = attr
VALUE             = value
issuer            = "$Issuer" "=" value
port              = "$Port" [ "=" <"> value <"> ]
```

Définition 15: Directive proxy-cookie

Pour la directive `Proxy-cookie`, nous avons également et pour les mêmes raisons supprimé les attributs `domain` et `path`. Et nous introduisons également et pour les mêmes raisons l'attribut `Issuer`.

Enfin, toujours par rapport au RFC nous ajoutons les règles d'utilisation suivantes:

Dans le cas d'une redirection automatique, le proxy DOIT préciser le ou les numéros de port pour lesquels le client doit renvoyer les proxy-cookies concernés.
Dans tous les cas, les proxy-cookies NE DOIVENT JAMAIS être relayés jusqu'au serveur d'origine.

4 Conclusion

Notre solution est extrêmement facile à implémenter au niveau du proxy (voir au chapitre 7 l'exemple de TomProxy), et l'impact sur les performances est tout à fait négligeable. Les documents peuvent en effet être encore manipulés comme des boîtes noires. Au niveau des en-têtes http, elles sont de toute façon traitées au niveau du proxy, puisque certains champs ont une influence sur le traitement des requêtes au niveau du proxy, et notre méthode ne demande que l'insertion d'un nouveau champ.

Au niveau du client, l'impact est plus conséquent, puisqu'il s'agit de proposer un nouveau style de cookie avec le stockage et la gestion que cela implique. Cependant, les proxy-cookies que nous proposons sont suffisamment similaires aux cookies classiques pour que le code soit réutilisable à 90%.

PROXY ET WEB SEMANTIQUE

Bénéfices pour les proxys avec l'enrichissement sémantique apporté par les différentes normes XML, les annotations et les ontologies

1 Introduction

Dans le chapitre 3, nous avons employé le terme "sémantique" dans sa signification classique (relatif au sens) le but étant d'extraire de la signification d'éléments disponibles dans le Web classique (par opposition au Web sémantique) afin d'étendre les dimensions accessibles à la perception du proxy. Dans ce chapitre, nous commencerons par redéfinir le terme sémantique dans son acception en ce qui concerne le Web sémantique. Nous définissons également une nouvelle qualité, l'engnose, comme l'extension de la perception au domaine de la connaissance.

Ensuite, nous présentons brièvement le Web sémantique et les technologies associées : XML, documents composés, annotations et ontologies. Pour chaque point, nous évaluerons l'impact et surtout le potentiel pour les proxys.

Enfin, pour les ontologies, nous exposons en détail la gestion de la connaissance qui est sous-jacente à ce domaine en insistant plus particulièrement sur la notion de partage de la connaissance au sein d'une communauté. C'est de ce point de vue que nous partons pour démontrer le rôle central que peuvent jouer les proxys dans le Web sémantique.

Il est difficile de trouver une définition admise par tous pour le *sémantique Web*¹³. Plus qu'une technologie ou un ensemble de normes, nous parlerons d'une vision proposée par Tim Berners-Lee. Dans une conférence, celui-ci propose¹⁴ :

- Le *Web* est la philosophie d'un espace de navigation avec des URI qui adressent des ressources
- *Sémantique* signifie qui peut être traité par une machine

On trouve aussi dans [73]: "Des données enrichies et interconnectées constituent la base du Web sémantique". L'auteur entend par *données enrichies* des informations accompagnées de méta-données, c'est-à-dire des données décrivant des données.

Plus prosaïquement, le Web sémantique propose l'extension des éléments du Web "classiques" par une structuration mieux conceptualisée (par exemple, séparation du contenu, des informations de présentation et des informations de structuration) et l'adjonction de méta-données.

Dans notre recherche de nouvelles dimensions pour la gestion du cache d'un proxy, tous les efforts actuels se fondant sur le principe de sémantique Web sont une aubaine. L'intention de rendre plus explicite la valeur sémantique des documents qui sont disséminés sur la toile nous permet, du côté du proxy, d'extraire plus facilement et de manière plus pertinente un certain nombre d'informations sur le plan qualitatif et

¹³ Pour s'en convaincre, il suffit de consulter les mailing listes sur <http://lists.xml.org/> et de constater les différentes interprétations données.

¹⁴ Propos extraits d'un résumé de la conférence donnée par Tim Berners-Lee et rapportés à l'adresse : <http://www.xml.com/pub/a/2000/12/xml2000/timbl.html>

quantitatif. Ceci est d'autant plus vrai au regard de la définition de "sémantique" donnée par Tim Berners-Lee.

Ces informations pertinentes ont deux sources : du côté des utilisateurs, l'introduction de la possibilité d'annoter des documents apporte des informations non seulement complémentaires sur le document lui-même, mais aussi sur la place subjective qu'occupe ce document dans la perception d'un utilisateur ou d'un groupe ; du côté de l'émetteur, la structuration des données au moyen des différentes normes rattachées à XML qui différencient clairement le contenu, la structure du contenu et la présentation, apportent également une lisibilité beaucoup plus grande sur la sémantique du document. De plus, XML introduit un enrichissement des liens entre les documents qui ne sont plus uniquement des pointeurs d'un point vers un autre comme dans HTML, mais des relations variées et hiérarchiques entre les différents éléments.

Nous avons déjà décrit la qualité plésiocentrique du proxy puisque c'est un élément de passage de l'intérieur vers l'extérieur d'une communauté (et réciproquement). Une autre qualité inhérente est, bien que ce soit un élément passif et transparent, la perceptivité de cet intermédiaire. Dans les proxys existants, c'est une perceptivité faible: relevés statistiques d'utilisation, propriétés apparentes des documents. Dans le chapitre trois, nous avons déjà étendu cette perceptivité aux relations qui lient les documents entre eux et à la position basée sur l'adressage. Avec le développement du Web sémantique, nous poursuivons un objectif similaire avec l'augmentation de la perceptivité aux méta-données et aux connaissances associées. A l'instar de l'empathie, qui est l'appropriation des sentiments de l'autre, nous définissons l'engnose comme l'appropriation passive des connaissances.

Pour un certain nombre de points, nous ne donnerons qu'une vue d'ensemble des différentes dimensions révélées dans le Web sémantique. Certains concepts sont encore à l'état d'élaboration et de recherche, par exemple XML, auquel viennent s'ajouter de nouvelles propositions ou encore les annotations pour lesquelles on trouve plusieurs propositions. Cependant, nous explorerons toutes les possibilités qui s'offrent dans ce domaine avec pour objectif soit l'engnose, soit la mise en place de services associés au Web sémantique en bénéficiant de la position plésiocentrique des proxys.

Pour cela, nous avons établi trois parties: XML, les annotations et la composition. Nous examinons dans ces trois domaines les techniques qui rentrent dans nos objectifs : amélioration du cache et engnose. La progression va de XML, qui permet de valoriser la sémantique du contenu; aux annotations, qui ajoutent la possibilité de décrire et d'ajouter de la sémantique; à la composition qui permet de re-structurer cette sémantique par découpage.

Dans la dernière partie, nous évaluons la position du proxy dans un environnement sémantiquement enrichi et par rapport à la gestion de la connaissance par les ontologies.

2 XML

Nous considérons que le Web sémantique est un sous-ensemble d'XML : un certain nombre de spécifications ou parties de spécifications sont motivées par cette vision. Mais le Web sémantique englobe également les normes XML dont il fait usage pour structurer et permettre la distribution de la connaissance [74].

2.1 Langage de structuration des données

XML propose d'une part une structuration généraliste des documents avec la possibilité de normaliser les structures publiées au moyen de DTD, qui peuvent être plus ou moins

largement normalisées. XML permet d'autre part la stricte séparation entre les données, la structure de ces données, et la représentation des données (au moyen de XSL).

XML couvre un vaste champ d'applications et de nouveaux horizons pour la distribution de l'information sur les réseaux. Nous avons choisi de ne pas explorer plus avant ce vaste domaine. De nombreux travaux de recherche¹⁵ s'y rapportent. Nous tenons cependant à souligner le bénéfice immédiat pour les proxys de disposer de données plus clairement structurées dans les documents qui transitent. Nous ne doutons pas qu'un certain nombre de travaux aboutis dans ce domaine ne manqueront pas d'être applicables pour l'enrichissement des caching proxys. Par exemple la séparation entre la présentation et les données permet de cacher des données identiques avec des présentations différentes.

2.2 *Sémantique des liens : XLink*

Dans sa première spécification du projet Web, Tim Berners-Lee propose quatre sortes de liens pour connecter les documents entre eux [6] (référence, référé, auteur(s), commentaire(s)). Cette distinction sera ensuite abandonnée dans la définition d'HTML par souci de simplification, mais elle revient dans les propositions de normes XML avec XLink. Le XML Linking Language (XLink) propose d'enrichir les liens entre les documents hypertextes du Web de manière significative pour apporter une réponse à la sobriété excessive des liens `` du langage HTML.

Dans la recommandation du W3C [75], on peut souligner, pour ce qui nous intéresse :

- Les liens ont un type : simple (très similaire aux liens HTML) ou étendu (supporte alors l'inclusion, les associations n-aires, les descriptions, etc.)
- Les liens ont une sémantique avec des attributs pour les décrire : role, arcrole et title qui sont autant d'informations pour l'utilisateur.

Ce dernier point est particulièrement intéressant puisqu'il apporte une dimension sémantique clairement préhensible aux liens qui connectent les documents entre eux, et ceci indépendamment des documents et de leur sémantique propre.

L'apport sémantique de ces attributs peut être mis en corrélation avec les rôles dans les ontologies. Par conséquent, l'utilisation de XLink permet de faire un pas en avant dans la direction des ontologies avec la possibilité de publier des rôles entre documents. Bien évidemment, il existe une distorsion entre l'ontologie utilisée, qui procède de concepts et de relations/rôles qui les lient, et les documents disséminés qui sont construits sur un ensemble de ces concepts. Cependant, la sémantique des liens entre les documents apporte un ordonnancement entre les documents et nous permet d'établir plus facilement les relations entre les concepts impliqués.

A partir des développements présentés au chapitre 3, et plus particulièrement concernant le web-graph, on comprend bien l'enrichissement que peuvent apporter ces nouvelles normes dans notre recherche de nouvelles métriques pour le caching. Une sémantique des liens connectant les documents nous permet naturellement d'enrichir la position relative des documents entre eux et de tenir compte de cette topologie pour la gestion du cache.

Considérons l'exemple d'un article publié par un laboratoire de recherche. Dans un document du laboratoire, cet article sera référencé par un lien de type "publie". Dans un index général de type Yahoo, le lien sera référencé par un lien sans type ou avec un type "index". Dans le premier cas l'utilisateur pourrait souhaiter remonter le lien s'il désire des informations sur les recherches du laboratoire, alors que dans le deuxième cas, il s'intéresserait uniquement au sujet de l'article.

¹⁵ Pour quelques pistes : <http://www.w3.org/XML/>, <http://www.xml.org/>

De plus, en cas de déficit sémantique (par exemple un document ne contenant que des images) on pourra étendre la valeur sémantique des documents référents.

3 Annotations

Comme nous l'avons déjà suggéré dans notre introduction historique sur le Web, il était au début question, pour la communauté de physiciens à laquelle était destinée l'outil WWW, de pouvoir annoter les documents publiés par des confrères. Dans cette optique, les premières propositions de Tim Berners-Lee incluait la possibilité de pouvoir annoter les documents. Avec la généralisation de l'utilisation du Web, cette fonctionnalité a rapidement [40] été abandonnée, car ingérable dans un contexte universalisé. On comprend aisément que l'anarchie la plus totale aurait alors rendu très rapidement ces annotations inutilisables, ou alors qu'il aurait fallu mettre en place des moyens considérables à des fins d'identification et de contrôle.

Des développements comme slashdot (code slash : <http://www.slashdot.org>) démontrent bien qu'il y a un intérêt pour des annotations ouvertes au public, ouvrant ainsi la porte aux développements collaboratifs très chers aux internautes. Il existe bien d'autres exemples de solutions particulières qui ont été mises en place pour permettre à ces derniers de partager des idées (sourceforge en est un autre excellent exemple). Mais tous ces efforts sont restés des solutions particulières qui ont été mises en place en différents points de la toile et qui ne répondent que partiellement aux ambitions globales des premières propositions, puisque chaque site implémente son propre système.

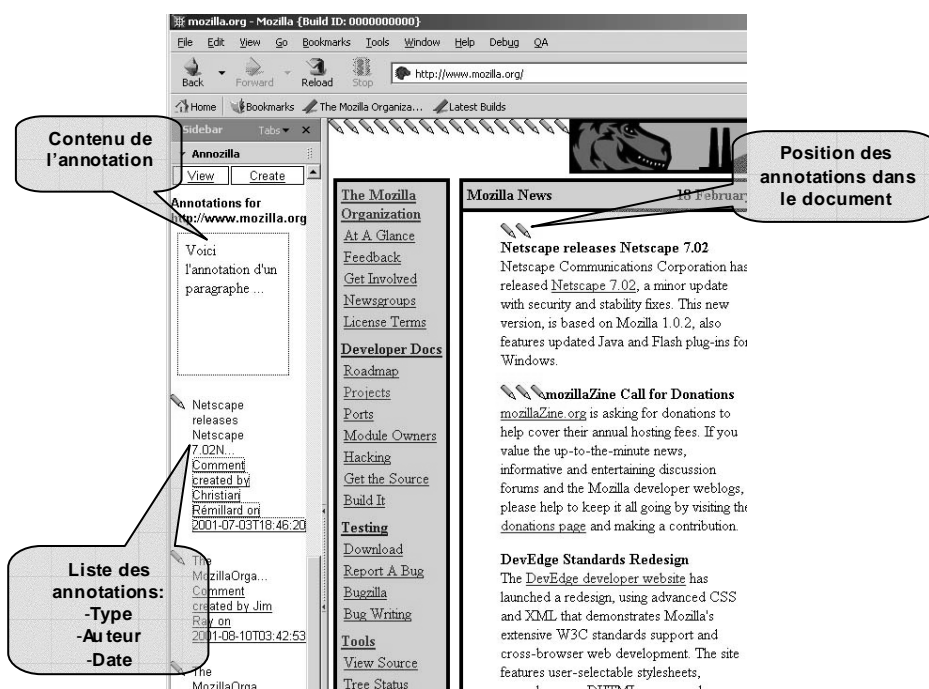


Figure 22 : Exemple d'utilisation des annotations

Les outils d'annotation permettent d'insérer des annotations dans des documents. On trouve un exemple d'utilisation d'un tel outil ci-dessus (Figure 22) avec Annozilla¹⁶. Celui-

¹⁶ Annozilla fait partie du projet Mozilla. Il vient s'intégrer dans le navigateur comme un plugin. Nous en reparlons plus loin (chapitre 7).

ci permet à l'utilisateur de créer de nouvelles annotations qui sont alors disponibles pour tout le monde. Dans l'exemple, on voit que l'outil affiche pour chaque document la liste des annotations qui lui sont associées et indique dans le document lui-même où sont situées ces annotations par l'insertion de petites icônes. Une fenêtre permet de visualiser le contenu de chaque annotation et on voit dans la liste que, pour chaque annotation, on dispose des informations suivantes : le type (commentaire, question, conseil, explication, etc.).

3.1 *Annotations et méta-données*

Les outils d'annotation permettent de lier des commentaires, des précisions, des corrections, etc. sur des documents existants. De ce fait, on assiste à un enrichissement collaboratif des documents par l'adjonction de données associées à tout ou partie du document.

Les annotations constituent donc des méta-informations qui viennent se greffer sur les documents existants et les enrichissent. Ces méta-informations peuvent être plus moins structurées, on a alors constitué des méta-données qui enrichissent la sémantique, au sens donné par Tim Berners-Lee, du document. Ces méta-données ajoutent aussi bien des propriétés que des informations complémentaires sur le document.

3.2 *Localisation*

Les outils d'annotations stockent ces informations complémentaires soit localement, auquel cas elles ne sont disponibles que pour un utilisateur¹⁷, mais le plus souvent sur un serveur partagé.

Il existe bien entendu des serveurs publics ouverts à toute la communauté Internet, mais de tels serveurs ne brillent pas par la pertinence de leurs annotations (voir ci-dessus, Figure 22, le contenu de l'annotation: "voici l'annotation d'un paragraphe") ! Le W3C¹⁸ propose un serveur: Annotea¹⁹[76, 77] et "... il est hautement encouragé pour les groupes de travail collaboratif d'installer leur propre méta serveur". Ce qui revient à dire : pour un groupe de travail et un domaine donné, il faut un serveur d'annotations indépendant.

Nous retrouvons donc la qualité plésiocentrique pour les serveurs d'annotations. Il nous paraît donc parfaitement légitime de chercher à faire coïncider ces deux services.

4 **Composition**

L'inclusion de parties externes dans un document existe déjà dans la norme HTML : il s'agit par exemple des images qui sont stockées dans des fichiers externes au document HTML. Cette disposition présente déjà des avantages considérables pour le caching. Prenons l'exemple d'un logo de société qui revient sur toutes les pages du site de cette compagnie, celui-ci ne sera caché qu'une seule fois pour tous les documents du site. Malheureusement, cet arrangement n'est disponible que pour un nombre limité de cas (les images, les sons, les fonds) et ne permet pas de disposer des éléments textuels en inclusion.

¹⁷ On pourrait envisager la possibilité de partager le même fichier entre plusieurs utilisateurs, mais on aurait alors un problème de concurrence d'accès. Comme ce problème n'est pas traité dans les projets existants, nous écartons cette solution, d'ailleurs peu élégante.

¹⁸ World Wide Web Consortium, <http://www.w3c.org/>

¹⁹ <http://www.w3.org/2001/Annotea/>

Toute fragmentation avec une granularité suffisante pour isoler toutes les parties récurrentes partagées par un ensemble de documents apporte de meilleures performances pour le caching.

Or, la tendance du Web est très nettement à la publication de documents dynamiques et personnalisés avec soit des portions de documents générées dynamiquement, soit des arrangements personnalisés d'informations. Prenons ainsi MyYahoo : chaque utilisateur, au travers d'une procédure de personnalisation, configure sa propre page d'accueil en choisissant des éléments dans une liste et la disposition de ceux-ci. Les informations disponibles (information météo, indice boursier, dépêches, etc.) sont en nombre important, mais limité. Et surtout, ces informations sont identiques pour tous les utilisateurs (avec des validités variées, ainsi les indices boursiers sont remis à jour toutes les 15 min.). On constate donc qu'aucun caching n'est possible avec ces documents, alors que toutes les parties pourraient l'être.

Prenons par exemple des éléments fixes : en-tête du document, pied de page, etc. Ces parties de documents sont souvent généralisées sur un même site Web (corporate identity) et le caching de ces sous-éléments serait donc possible et judicieux.

Si on examine le cas d'un document d'ensemble dynamique, mais constitué de références pour inclusion d'éléments standard, on voit bien que la plus grande partie du document pourrait être efficacement cachée, alors qu'il n'en est rien à l'heure actuelle.

On trouve des solutions partielles dans les techniques existantes. Ainsi, les éléments HTML `<frameset>` (balise HTML qui permet de subdiviser un document en sous-documents) proposent un début de solution qui manque malheureusement de flexibilité, au point de ne pas avoir été adoptée dans les usages. Une autre solution est apparue avec les éléments `<DIV>` et `<LAYER>` (balise HTML qui permet d'insérer un document dans un autre) ou des bricolages javascript. Ces dernières possibilités sont effectivement utilisées sur des sites comme `cnn.com`, mais elles manquent de standardisation, de clarté et ne procèdent d'aucune systématique.

De manière générale, il serait souhaitable de pouvoir fragmenter les documents dynamiques jusqu'à obtenir une granularité qui délimite tous les éléments statiques possibles.

4.1 *Edge Side Includes*

Une première tentative mise en avant a été proposée par plusieurs acteurs commerciaux, notamment Oracle et Akamai : la norme ESI (Edge Side Includes) [78]. Cette proposition a été spécifiquement conçue pour le caching de pages dynamiques avec des données issues d'une base de données, en permettant la fragmentation des documents en partie statiques et partie dynamiques. Le proxy-cache reçoit donc un document composé d'éléments et, surtout, les références aux parties contiennent également des indications sur la politique de cache à appliquer. ESI est donc spécifiquement orienté caching de documents pour un site exploitant une base de données. Oracle propose ainsi une architecture CDN qui exploite intensivement ce procédé pour accélérer l'accès aux pages dynamiques [79].

4.2 *XML*

On trouve dans la spécification d'XML la norme XInclude [80] qui permet également au travers de la spécification XPointer [81] de décrire des éléments à inclure dans un document. Sans nous étendre sur cette norme simple en soi, mais compliquée par les concepts XML qui entrent en jeu (surtout au niveau du framework xpointer), nous noterons que par rapport à la proposition ESI, XInclude ne propose aucune directive ni indication explicite pour le caching. Les pointeurs XML utilisés pour décrire les éléments

devant être inclus, permettent sans doute d'ajouter des attributs porteurs d'une sémantique destinée au cache, mais rien n'a été normalisé dans ce sens-là. Un avantage en revanche certain de XInclude est la souplesse dans les références, puisqu'on peut adresser aussi bien des documents que des parties de documents avec un mécanisme très complet dans ce sens.

Notons que pour le moment, rien n'ayant été défini à ce sujet, les caching proxys induisent plutôt des problèmes avec des serveurs comme Cocoon (serveur d'applications XML du consortium Apache) cachant des pages dont certaines parties sont pourtant dynamiques. On obtient d'ailleurs aussi des effets de bord très agaçants avec les stylesheets (XSL) qui persistent dans le cache du navigateur alors qu'ils ne sont plus valides.

4.3 *Extension à d'autres médias*

Nous n'avons présenté jusqu'ici que le cas des documents textes. La même règle d'affinement s'applique pour obtenir une granularité permettant de distinguer les parties statiques des parties dynamiques et les éléments récurrents. Le problème est bien entendu rendu plus compliqué pour des médias comme les images, les sons et la vidéo, avec notamment des effets de superposition difficiles à rétablir. On notera que les techniques de vectorisation répondent à ce besoin de fragmentation en éléments.

Nous proposons d'examiner comme exemples deux techniques de vectorisation qui apportent une fragmentation permettant un meilleur caching.

4.3.1 *Multimédia*

Le terme même de "multimédia" est relativement imprécis. Le sens premier est la propriété de contenir plusieurs formes de données : audio, vidéo, texte, etc. Dans son acception plus usuelle, le multimédia désigne tout ce qui dépasse le simple document texte. On attribue ainsi la qualité de multimédia à des documents pourtant mono média comme une vidéo ou un fichier audio.

Dans le cas des objets mono média, la composition peut être simplement le découpage linéaire dans le temps d'un objet en segments. On offre ainsi la possibilité d'accéder à une portion de l'objet rapidement, sans avoir à télécharger le tout. Le protocole HTTP 1.1 permet en effet en théorie l'accès à une portion de fichier, mais cette fonctionnalité n'est pas supportée par les caching proxys. Or, une fois de plus, l'affinement de la granularité devrait pourtant permettre de meilleures performances.

Dans le domaine de la vidéo et encore plus dans les animations graphiques, on trouve en plus du découpage en séquences, le découpage en éléments : fonds, objets mobiles, éléments sonores, synthé²⁰. Ces éléments sont récurrents et pourraient donc être cachés indépendamment de l'ensemble [82].

4.3.2 *SIG*

Le cas des SIG (Systèmes d'information géographiques) est plus particulièrement intéressant. La mise à disposition de telles informations sur le Web est en effet en pleine expansion, et souvent avec des performances passablement dégradées ainsi qu'une très grosse consommation en bande passante. Or, ces systèmes sont à la base constitués d'informations vectorielles recomposées avant d'être envoyées au client sous forme d'image (pour exemple : <http://map.epfl.ch/>²¹).

²⁰ Titre incrusté dans l'image. Utilisé systématiquement dans les téléjournaux pour indiquer le nom de l'interlocuteur à l'écran.

²¹ Plus d'informations: <http://sawwww.epfl.ch/SIC/SA/publications/FI02/fi-7-2/7-2-page1.html>

Pour le proxy, chaque image ainsi transmise à un client est un nouvel objet dynamique et volumineux qui n'est donc pas du tout caché, alors qu'en réalité il s'agit d'un ensemble d'informations vectorielles de petite taille et parfaitement statiques.

5 Gestion de la connaissance: les ontologies

"Il y a plus de trente ans, Ed Feigenbaum, lauréat du ACM Turing Award, annonçait une révolution dans l'informatique sous l'appellation: *Knowledge is Power*²². [...] Cette approche [le Web sémantique] pourrait aussi être intitulée: *Knowledge is Power*. [...] Ce nouveau paradigme [le Web sémantique] apparente la connaissance à la puissance en transit dans la grille électronique. [...] En bref, nous voyons approcher le jour où l'encapsulation prudente de la connaissance dans des applications spécifiques à certains domaines se transforme en une omniprésence de sources de connaissances reliées les unes aux autres dans une large toile de connaissances distribuées.", tiré de [83].

La prolifération du Web a soulevé l'enthousiasme. Mais on constate aujourd'hui une certaine déception par rapport à la valeur de l'information, à la pertinence des documents publiés et surtout à la possibilité de trouver une information adéquate : quantité n'est pas qualité. Le grand défi de la prochaine génération du Web, le Web sémantique, se situe dans la valorisation des informations pour un meilleur partage de la connaissance au travers des réseaux. Dans ce chapitre, nous allons d'abord analyser les modes et les substrats sur lesquels ces échanges de connaissances s'établissent. Les connaissances s'échangent en fonction de conventions culturelles et communautaires et avec le support d'un certain formalisme. Ensuite, nous verrons comment la formalisation par le biais des ontologies ainsi que l'utilisation des annotations donnent un support à l'échange d'informations et quel rôle les proxys peuvent tenir dans ce contexte.

5.1 Partage de la connaissance

Le titre même de [84] : "Towards The Semantic Web, Ontology-Driven Knowledge Management", suggère explicitement que le Web sémantique et l'adjonction des ontologies impliquent la gestion de la connaissance. Mais la volonté de gérer cette connaissance en réseau est la manifestation archétypique pour l'humanité de partager son savoir. Bien que l'on évoque avec enthousiasme le village global, la malédiction de Babel n'a pas encore été surmontée, et il existe toutes sortes de barrières qui limitent la compréhension dans la communication des connaissances acquises.

Nous allons donc examiner les barrières liées à la constitution des communautés ainsi qu'au partage des conventions au sein de ces communautés, et comment les ontologies permettent un meilleur partage de la connaissance.

5.1.1 Communauté culturelle

Edward T. Hall écrit dans son introduction à [85] : "... les individus appartenant à des cultures différentes non seulement parlent des langues différentes mais, ce qui est sans doute plus important, *habitent des mondes sensoriels différents*." Ainsi, non seulement la langue, mais tout un ensemble de perceptions, de coutumes, de codes constituent des dimensions que partage un groupe. La définition de groupe culturel est évidemment complexe, et surtout procède de limites floues. Un peuple constitue ainsi un groupe culturel identifié, une région linguistique ; on en compte par exemple quatre en Suisse alors que d'autres pays ne comptent qu'une seule communauté linguistique. Comme Hall le démontre dans son ouvrage, les dimensions qui déterminent les groupes sont

²² La puissance vient de la connaissance.

nombreuses et entremêlées. Elles définissent des groupes aussi bien au niveau continental qu'à la taille de l'entreprise.

Comme nous l'avons décrit plus haut avec la propriété plésiocentrique des proxys, nous constatons une certaine adéquation entre la disposition de ceux-ci et la définition de ces communautés, même si elle n'est pas vérifiée de manière systématique. Par exemple, les universités suisses proposent en général un service de caching proxy dans une hiérarchie qui est chapeautée par un proxy suisse, proposé lui-même par l'opérateur Internet académique suisse (SWITCH). Ainsi, au niveau académique en tout cas, nous constatons une structure qui adhère au moins partiellement au découpage culturel national.

5.1.2 *Partage des conventions*

L'héritage culturel d'une communauté induit directement un certain nombre de conventions partagées par le groupe. Ces conventions ont un rôle extrêmement important dans la communication et les échanges d'informations au sein de la communauté qui les partage comme le démontre [86]. Dans ce même ouvrage, on constate que le partage explicite de ces conventions, et même le renforcement du conventionnement des valeurs partagées, permet une amélioration du rendement du groupe.

Au niveau linguistique, on trouve des groupes qui se définissent par des expressions nommées *idiolectes* [87]. L'usage et le sens de ces expressions sont propres à une communauté restreinte et ces codes perdent toute leur sémantique au-delà du groupe concerné. Ainsi trouve-t-on des *idiolectes* familiaux, professionnels, etc.

Un exemple de conventions différentes et même contradictoires, pourtant dans une même communauté, est illustré par les différentes modélisations. Ainsi en UML, les cardinalités sont à l'inverse de ce que l'on trouve dans la modélisation entité association [88].

De nouveau, la situation plésiocentrique des proxys en fait un acteur privilégié au sein du groupe pour l'établissement et le partage de ces conventions.

5.1.3 *Partage de la connaissance et des ontologies*

Une fois des conventions culturelles établies au sein du groupe, on peut alors construire efficacement un réservoir de connaissances et structurer cette connaissance en se basant sur ces conventions.

On est encore loin de pouvoir considérer le village global comme une seule et même communauté, même si certaines dimensions, par exemple la *netiquette*²³, sont universelles. Les normes proposées pour la définition de domaines d'activités vont en général aussi dans le sens de proposer des conventions universelles (V-Card, DTD publiées pour différents domaines, etc.), mais la localisation n'est pas prête de s'effacer.

Par conséquent, les groupes, les communautés, auront toujours besoin de partager des représentations communes de la connaissance qu'ils partagent.

Les ontologies qui sont une formalisation de la structuration des connaissances se veulent plus universelles, mais elles ne pourront pas non plus être complètement universalisées avant longtemps. La langue dans laquelle sont transcrites les ontologies en constitue déjà un bon exemple.

5.2 *Annotations et ontologies*

« La raison pour laquelle les ontologies deviennent si populaires tient dans la promesse : "une compréhension commune et partagée d'un domaine qui peut être communiquée entre les gens et les systèmes applicatifs" », tiré de [84]. Dès le moment où l'on ouvre la

²³ Code de conduite sur Internet.

possibilité aux utilisateurs d'annoter des documents, il devient souhaitable que ces annotations procèdent d'une structuration qui permette d'une part un échange de points de vue entre personnes, d'autre part un enrichissement de la valeur sémantique du document annoté (voir exemple plus haut).

Les annotations deviennent ainsi un moyen d'attacher plus que de simples textes, mais véritablement des méta-données qui viennent s'intégrer dans le Web sémantique: "ces méta-informations définissent de quoi il s'agit dans une forme analysable par une machine"[84]. De plus, si un utilisateur annote un document, on peut espérer qu'il s'est d'abord attaché à comprendre le contenu qu'il commente, qu'il a une vision relativement claire de lui-ci et qu'il est capable de le situer dans son univers de concepts. Etant donnée cette appropriation intellectuelle de l'utilisateur, pourquoi en effet ne pas profiter de l'effort qu'il fait pour lui demander de disposer son commentaire dans une forme structurée et de le classer dans une ontologie pour un domaine donné (en l'occurrence le sien !).

Evidemment, on pourra objecter que l'auteur est sans doute la personne la plus à même de situer son document, mais ce sera dans son propre univers, autrement dit dans l'ontologie qu'il se sera construite lui-même, de la même manière qu'en littérature ce ne sont pas les auteurs (ou rarement !) qui proposent l'analyse de leurs propres œuvres. Ils peuvent en revanche procéder à l'herméneutique d'un autre auteur (à l'instar de Sartre commentant Camus, ou Paul Valéry divers écrivains), l'auteur propose et les lecteurs disposent. C'est la raison pour laquelle les annotations sont détachées du document, elles ne suivent pas le même processus de publication.

5.3 *Proxy et localisation*

Dans le cas du Web, et surtout de son universalité, le problème est encore plus pertinent : différents groupes d'utilisateurs ou même des individus ne structurent pas leur univers de la même manière et ne partagent pas les mêmes valeurs. Ils ne proposeront donc pas de facto la même classification pour les mêmes documents. Pour reprendre notre parallèle avec la littérature, pour un même ouvrage, on retrouve souvent dans ses analyses des points de vue différenciés voire contradictoires. Comme les annotations et la classification procèdent d'une certaine localisation (individu ou groupe), quel meilleur endroit que le proxy pour les gérer ? Nous proposons comme exemple l'utilisation d'un vocabulaire spécialisé dans les annotations.

Il subsiste cependant un problème : la disposition physique (géographique) des proxys et des infrastructures réseau n'est pas forcément en parfaite corrélation avec la couverture des communautés. On trouve aussi des communautés dispersées. Il se pose alors le problème d'adaptabilité à la morphologie du groupe. La granularité du découpage induit par la répartition des serveurs proxys ne correspond par forcément à la cellule des groupes qui s'établissent. Pour reprendre l'exemple cité plus haut, s'il existe un proxy national qui englobe tous les groupes académiques, il n'existe pas de proxy national global ou par région linguistique. De même, si l'EPFL dispose d'un cache pour toute l'école, il n'existe pas de cache pour les unités, à savoir les différents labos, instituts, départements, etc. qui s'apparentent pourtant à des communautés partageant toutes sans distinction un même espace de caching.

Cependant, le fait de disposer de profils utilisateurs, élaborés au chapitre 4, nous permet de définir des groupes indépendamment de la topologie physique des caches. En effet, on peut associer au profil l'appartenance à un groupe. Ainsi, nous pouvons simplement et complètement nous adapter à la morphologie des groupes qui se constituent, pour autant que la granularité soit plus faible que la couverture du proxy (c'est-à-dire que l'ensemble de la communauté se trouve derrière le proxy). Dans le cadre de groupes plus dispersés

que les infrastructures couvertes par un proxy, nous pouvons envisager une collaboration inter-proxy qui établisse un schéma de mobilité.

5.4 Engnose

Nous avons donc la possibilité, avec les (méta-)informations mises à disposition dans le Web sémantique, notamment au travers des annotations, de rendre le proxy perceptif à une classification par concepts et domaines des documents qu'il "voit" passer. Nous avons défini en introduction l'engnose comme la capacité à pouvoir s'appropriier des connaissances. Cette appropriation passive s'inscrit ici dans la capacité des proxys à pouvoir bénéficier des méta-données associées aux associations, méta-données qui viennent s'inscrire dans une structuration ontologique de la gestion de la connaissance.

6 Conclusion

Les proxys jouent le rôle d'intermédiaire entre l'intranet et Internet, entre une communauté et le monde extérieur. Ils sont également au centre de la communauté et donc virtuellement au centre d'un ensemble de connaissances partagées par ses utilisateurs, à condition que ces connaissances puissent être dans une représentation matérialisée perceptible pour le proxy.

Plus loin, en plus d'être un acteur passif de captation des connaissances en transit, le proxy peut également servir de plateforme de rassemblement des connaissances, des conventions et de la structuration de ces connaissances dans des ontologies. On s'oriente donc non seulement vers une extension perceptive du proxy, mais également vers la mise à disposition d'une plateforme pouvant supporter des services associés à la gestion de la connaissance.

Les proxys, par leur qualité plésiocentrique, sont au cœur de la régionalisation comme le démontrent d'autres travaux comme [89]. Ainsi le $\gamma\upsilon\omega\theta\iota\ \sigma\alpha\upsilon\tau\acute{o}\nu$ ²⁴ de Socrate devient $\gamma\upsilon\omega\tau\epsilon\ \upsilon\mu\acute{\alpha}\varsigma\ \alpha\upsilon\tau\acute{o}\upsilon\varsigma$ ²⁵ où les proxys ont un rôle privilégié à jouer dans cette perspective.

²⁴ Connais-toi toi-même. Inscription sur le frontispice de la première université fondée par Platon

²⁵ Connaissez-vous vous-mêmes.

PROXYS MÉTIER

Mise à profit de l'engnose pour l'élaboration de services prenant en compte la disponibilité de la connaissance

1 Introduction

Dans le chapitre précédent, nous avons défini l'engnose dans l'intention de rendre notre proxy plus perceptif notamment aux connaissances que véhiculent les documents de la toile. Nous avons également examiné la forme et la disposition de ces connaissances pour une communauté partageant un proxy. Nous avons suggéré que cette augmentation de la perceptivité devait permettre au proxy aussi bien d'améliorer la qualité des services en mode passif (essentiellement le caching) que de permettre l'élaboration de nouveaux services plésiocentriques.

Dans ce chapitre, nous allons aborder le problème de l'adaptation et de la spécialisation du caching proxy en fonction de l'utilisation, d'où le terme de proxy métier. Nous allons d'abord présenter une adaptation du caching avec une solution de cache métier. Pour cela, nous définissons un système de bascule dans le domaine (métier) courant en se basant sur la dimension ontologique perçue au travers de la navigation. Nous exploitons ainsi l'engnose définie auparavant.

Dans la deuxième partie, nous prendrons un cas d'étude concret, SIRANAU, et nous détaillerons l'utilisation qui a été faite des proxys dans ce contexte.

Le but de ces deux démarches est de démontrer comment l'utilisation des intermédiaires représente une solution intéressante à certains problèmes.

2 Caches métier

Nous avons établi dans le chapitre précédent qu'un proxy pouvait servir plusieurs communautés distinctes et que ces communautés possédaient des conventions et des connaissances propres. De plus, elles ne partagent pas forcément les mêmes intérêts, ce qui sous-entend des utilisations d'informations différentes. Nous regroupons ces intérêts dans des domaines liés à des ontologies et nous avons proposé de rendre accessible ces ontologies au proxy. Cela signifie que nous avons donc mis à la disposition du proxy les informations nécessaires pour être capable d'identifier des domaines.

Comme le concept des annotations est revenu à l'ordre du jour en pleine effervescence pour le Web sémantique, il s'est donc immédiatement associé aux ontologies. Puisque l'utilisateur prend le temps et la peine d'annoter un document, il est plus que pertinent de considérer qu'il a une bonne connaissance du contenu de ce document. Par conséquent, on peut supposer qu'il est capable de situer ce document par rapport à ses domaines d'activité et plus précisément de le classer dans une ontologie associée à son domaine d'activité. Il est donc raisonnable de solliciter l'utilisateur qui enregistre une annotation sur un document, de lui demander de définir la position qu'occupe ce document dans une ontologie.

Nous allons, dans cette partie, exposer comment identifier le domaine courant et comment l'exploiter dans la gestion du cache. Pour cela, nous définissons d'abord le concept de cache métier.

Définition 16: Un cache métier est un cache qui peut prendre en compte dans sa gestion le domaine courant de l'utilisateur.

Et nous définissons l'opération qui consiste à évaluer la valeur ontologique courante pour basculer dans un domaine.

Définition 17: L'onto-plexing désigne le processus de bascule dans un domaine courant en fonction des ontologies à disposition (connues) et de la valeur sémantique et/ou ontologique des ressources courantes.

2.1 Nécessité de disposer de caches métier

Nous avons donc montré qu'il existait des groupes qui partagent des intérêts, des connaissances et des conventions différentes. Ces groupes ont évidemment des usages et des besoins différents, ils ne s'intéressent pas aux mêmes domaines et ne consultent par conséquent pas les mêmes documents.

Il est par contre très probable que des groupes différents partagent physiquement un même caching proxy. Et ces groupes rentrent donc en concurrence pour l'utilisation du cache. Les études (voir chapitre 2) sur les proxys se focalisent sur les mécanismes du caching proxy au niveau macroscopique, en l'occurrence le comportement de l'ensemble des utilisateurs d'un proxy. Pourtant ayant démontré auparavant que la constitution de groupes est inhérente à une activité humaine diversifiée, il apparaît nécessaire de s'intéresser à une granularité plus fine, c'est-à-dire au niveau d'un sous-groupe d'utilisateurs ou de l'utilisateur lui-même.

Pour reprendre l'exemple de notre Ecole qui accueille par exemple un certain nombre d'étudiants de la partie italophone de notre pays, nous trouvons donc une communauté minoritaire de langue italienne.

2.1.1 Distribution de Zipf

Plusieurs travaux [61, 62] montrent que la distribution des accès aux documents sur le Web pour une communauté d'utilisateurs, suit une loi de Zipf: le document de rang n à une probabilité de $\frac{1}{n}$ d'être accédé (ou loi similaire proposée: $\frac{1}{n^\alpha}$, voir également chapitre 2).

De cette probabilité il découle directement que la distribution des accès sur les documents suit une fonction similaire et que si l'on a:

N_i le nombre d'accès au document de rang i (avec $i \in [1..k]$), on aura:

$$N_2 = \frac{N_1}{2}, N_3 = \frac{N_1}{3}, \dots, N_k = \frac{N_1}{k}$$

La question qui se pose alors est la façon dont se combinent les distributions des accès aux documents pour deux communautés qui partagent le même cache physiquement.

Nous ne disposons pas de données statistiques suffisantes pour pouvoir inférer une règle. De plus, la distribution de Zipf est une approximation qui se révèle adéquate pour décrire les accès vus par un cache dans le cas de fréquentations très importantes (les études faites dans [61] portent sur des périodes et surtout des populations importantes). Mais il semble

acceptable de considérer qu'en réduisant le champ d'observation, la distribution reste similaire.

Par conséquent, il est toujours raisonnable de considérer que les distributions des accès pour deux sous-groupes d'utilisateurs d'un même cache physique suivent individuellement et approximativement aussi une distribution de Zipf.

Reprenons l'exemple de nos deux communautés linguistiques : francophone et italienne et les quatre documents les plus visités par ces deux communautés (certains documents étant éventuellement en commun, nous avons entre 4 et 8 documents):

	Rang 1	Rang 2	Rang 3	Rang 4
Francophones	50'000	25'000	16'666	12'500
Italophones	15'000	7'500	5'000	3'000

Nous avons bien pour ces deux suites indépendantes une progression qui convient à la loi de Zipf.

Nous allons maintenant nous intéresser à la combinaison de ces statistiques d'utilisation, comme ce serait le cas pour un caching proxy partagé par les deux communautés.

Au niveau macroscopique, la distribution de fréquentation des documents peut résulter de combinaisons variables de fréquentation des sous groupes. Nous pouvons cependant illustrer deux cas extrêmes :

C_1, C_2 : le sous-ensemble des documents fréquentés par la communauté 1 resp. 2.

- Soit les ensembles sont disjoints:

$$C_1 \cap C_2 = \emptyset$$

Dans ce cas, nous obtiendrons au niveau macroscopique la distribution suivante en reprenant l'exemple ci-dessus:

	Rang 1	Rang 2	Rang 3	Rang 4	Rang 5	Rang 6	Rang 7	Rang 8
C_1+C_2	50'000	25'000	16'666	15'000	12'500	7'500	5'000	3'000

Nous avons une distribution qui ne suit plus exactement la loi de Zipf mais reste très similaire.

- Soit les ensembles sont confondus:

$$C_1 \cap C_2 = C_1 = C_2$$

et nous obtenons:

	Rang 1	Rang 2	Rang 3	Rang 4
C_1+C_2	65'000	32'500	21'666	16'250

Le résultat est clairement exactement conforme à la loi de Zipf.

- Entre ces deux extrêmes il existe de nombreuses combinaisons possibles. Par exemple:

	Rang 1	Rang 2	Rang 3	Rang 4
	$C_1(k_1)+C_2(k_2)$	$C_1(k_2)+C_2(k_6)$	$C_1(k_3)+C_2(k_5)$	$C_1(k_5)+C_2(k_4)$
C_1+C_2	52'100	27'500	19'666	13'000

Cas aussi idéal par rapport à la loi de Zipf.

De ces exemples, on constate surtout que le premier cas est extrêmement défavorable à la communauté minorisée. Si par exemple notre cache ne conservait que les 4 documents les plus fréquentés (politique LFU), les documents en langue italienne seraient systématiquement exclus du cache.

En conclusion, si au niveau macroscopique, le rendement peut sembler optimal, il peut très bien se révéler catastrophique en réduisant le point de vue à l'une ou l'autre communauté sortie de l'ensemble des utilisateurs du proxy.

2.1.2 Cache métier

Le partitionnement du cache en cache virtuel a déjà été présenté plus haut dans ce travail (voir chapitre 2). Dans les travaux présentés dans [51, 53], l'espace physique du cache est partitionné verticalement dans le temps, c'est-à-dire qu'un document évincé du niveau 1 est alors caché dans le niveau 2, et ainsi de suite jusqu'au niveau n à partir duquel il est définitivement sorti du cache. Nous proposons ici de rajouter une dimension : le domaine de connaissance auquel se rattache le document. Nous pouvons ainsi proposer un partitionnement horizontal du cache : par domaine.

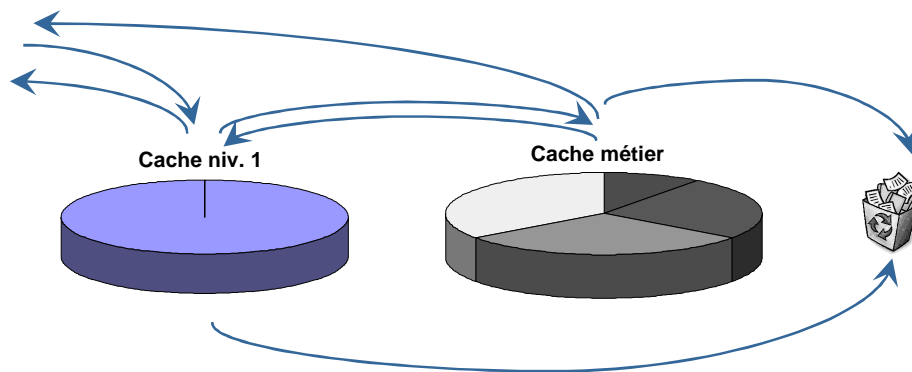


Figure 23 : Cache à étages et partitions métier

Comme nous l'avons dit plus haut, les ensembles de documents visités par deux communautés distinctes ne sont pas forcément disjoints, ceci dans des proportions tout à fait variables. De plus, il existe des documents généraux qui peuvent difficilement être attribués à tel ou tel domaine, notamment pour les documents qui sont des index vers d'autres documents. Il est par exemple difficile de déterminer si la page d'entrée du site Yahoo appartient plus au domaine des botanistes qu'à celui des anthropologues !

Nous proposons donc de combiner les deux dimensions : verticalement, plusieurs caches successifs avec différentes politiques de gestion, et appliquer à un des niveaux un découpage (nous choisirons a priori le deuxième niveau) par domaines d'activité.

En reprenant les résultats obtenus dans [51], nous appliquerons ainsi une politique LFU au premier niveau avec d'excellents résultats pour les documents généralement populaires. Au niveau deux, nous appliquerons alors une politique LFU par partitions pour les différents domaines.

Des plus, les traitements associés au partitionnement par domaines entraîneront des traitements plus lourds. Avec cette architecture, nous disposons d'un premier niveau de cache immédiat rapide et qui a prouvé son efficacité, alors que le deuxième niveau pourra être traité par un processus détaché en background.

2.2 Onto-plexing

Maintenant que nous avons défini les caches métier, nous allons proposer un système qui permette leur exploitation en proposant un mécanisme de bascule automatique dans le domaine courant, sans que l'utilisateur ait à intervenir.

En effet, si l'opportunité rapportée ci-dessus de pouvoir créer des groupes d'utilisateurs à la demande permet de se conformer à la réalité, l'effort pour maintenir ces groupes peut cependant se révéler rapidement fastidieux. De plus, un même utilisateur peut appartenir à plusieurs groupes distincts et partager ainsi plusieurs domaines d'intérêt en passant de l'un à l'autre au cours du temps.

A partir du moment où nous avons concentré dans une même application plésiocentrique les services de caching, d'annotation et de classification de la connaissance par ontologie, il devient alors évident d'exploiter l'ensemble de ces informations pour proposer une automatisation de la détermination du domaine dans lequel l'utilisateur est actif au cours de sa navigation.

Pour cela, nous allons définir un certain nombre de concepts: l'isotopie sémantique étendue, l'isotopie ontologique et enfin le scoring isotopique. Tous ces concepts nous permettent alors de mettre en place ce système de bascule automatique ou plus précisément d'onto-plexing automatique.

2.2.1 *Isotopie sémantique étendue*

On trouve dans [90] la définition suivante pour l'isotopie sémantique : "... la récurrence syntagmatique du même sème²⁶ ou groupement de sèmes. La relation d'identité entre les occurrences du sème ou du groupement sémique entraîne des relations d'équivalence entre les sémèmes (lexèmes) qui les comportent: des unités polysémiques peuvent être rendues monosémiques par la relation d'isotopie. Dans l'exemple de A.J. Greimas : 'belle soirée – et quelles toilettes', l'isotopie de la 'réception' amène à lire les toilettes comme 'robes' et non comme 'cabinets' ". Pour chaque lexème, son isotopie recouvre donc l'ensemble des synonymes ou expressions équivalentes dans un contexte donné et nous désignons cet ensemble:

$I(l/C) = \{l_1, l_2, \dots, l_k\}$ avec l un lexème et C le contexte dans lequel on se place.

Nous définissons également les ensembles antonymes, hyperonymes (sur classe) et hyponymes (sous-classe):

$A(l/C) = \{l_1, l_2, \dots, l_k\}$

$Her(l/C) = \{l_1, l_2, \dots, l_k\}$

$Ho(l/C) = \{l_1, l_2, \dots, l_k\}$

Nous définissons alors l'isotopie sémantique étendue pour un lexème par rapport à un contexte:

$I^+(l/C) = \{I(l/C) \cup Her(l/C) \cup Ho(l/C), A(l/C)\}$

On trouvera en linguistique la même notion dans le terme de champ dérivationnel.

2.2.2 *Isotopie ontologique*

En reprenant l'exemple introduisant le paragraphe précédent, on comprend bien que la sémantique varie en fonction du contexte. Nous allons donc prendre en compte le contexte, en l'occurrence en nous basant sur les ontologies, pour affiner la perception sémantique.

Nous définissons l'isotopie ontologique comme la composition des isotopies sémantiques étendues de tous les termes apparaissant dans cette ontologie. En linguistique [90], le terme de champ conceptuel est relativement équivalent à cette définition.

Ainsi, pour une ontologie nous avons l'isotopie de cette ontologie définie par:

²⁶ La plus petite unité du langage qui a du sens.

Soit $L = \{l_1, l_2, \dots, l_n\}$ l'ensemble des lexèmes de cette ontologie et C le contexte défini pour le domaine de cette ontologie:

$$I^{o,c} = \bigcup_{i=1}^n I^+(l_i / C)$$

$$I^{o,c} = \bigcup_{i=1}^n \{I(l_i / C) \cup Her(l_i / C) \cup Ho(l_i / C), A(l_i / C)\}$$

$$I^{o,c} = \left\{ \bigcup_{i=1}^n I(l_i / C) \cup \bigcup_{i=1}^n Her(l_i / C) \cup \bigcup_{i=1}^n Ho(l_i / C), \bigcup_{i=1}^n A(l_i / C) \right\}$$

Ainsi, l'isotopie ontologique nous donne le lexique complet de lexèmes associés positivement ou négativement au domaine décrit par cette ontologie.

2.3 Application aux proxys

En partant du principe que notre application proxy a accès aux ontologies définies ou utilisées par notre communauté d'utilisateurs (chapitre 5), nous pouvons pour chaque ontologie construire son isotopie. A partir des lexiques ainsi obtenus, un rapide parcours des lexèmes contenus dans les documents traversant le proxy nous permet d'évaluer l'adéquation entre une ontologie et le document en question.

2.3.1 Distance isotopique

Pour un document, nous extrayons l'ensemble des lexèmes (par extraction des token/lexèmes à l'exclusion des éléments de structuration et de présentation de l'information en utilisant des bibliothèques comme Sax par exemple):

$$TL = \{l_1, l_2, \dots, l_m\}$$

Nous définissons la distance d'un lexème l' à l'isotopie sémantique étendue $I^+(l/C)$ d'un lexème l : $d_{I^+(l/C)}(l')$

$$\text{avec } \begin{cases} \text{si } l' \in I(l/C) \cup Her(l/C) \cup Ho(l/C) & \text{alors } d_{I^+(l/C)}(l') = 1 \\ \text{si } l' \in A(l/C) & \text{alors } d_{I^+(l/C)}(l') = -1 \\ \text{sinon} & d_{I^+(l/C)}(l') = 0 \end{cases}$$

Remarque: la distance est réflexive.

Nous définissons la distance d'un lexème l à l'ontologie o : $D_o(l)$ avec $L = \{l_1, l_2, \dots, l_n\}$ les lexèmes de cette ontologie:

$$\begin{cases} \text{si } l \in \bigcup_{i=1}^n I(l_i / C) \cup \bigcup_{i=1}^n Her(l_i / C) \cup \bigcup_{i=1}^n Ho(l_i / C) & \text{alors } D_o(l) = 1 \\ \text{si } l \in \bigcup_{i=1}^n A(l_i / C) & \text{alors } D_o(l) = -1 \\ \text{sinon} & D_o(l) = 0 \end{cases}$$

Nous définissons enfin la distance d'un document T à une ontologie comme la somme des distances de chacun des lexèmes extraits du document T à l'ontologie pour un contexte donné:

$$S_{o,c}(T) = \sum_{l_i \in T} D_{o,c}(l_i)$$

Nous avons défini des distances, et plus particulièrement la distance d'un document à une ontologie, qui vont nous permettre d'évaluer si un document est proche ou non d'une ontologie.

Cette opération semble tout à fait vraisemblable. Si l'on se réfère au fonctionnement d'outils comme htdig²⁷, on se convaincra aisément qu'il est possible d'extraire tous les lexèmes d'un document donnée avec un temps de calcul acceptable.

2.3.2 *Autodétermination du domaine actuel de l'utilisateur*

À partir du moment où nous sommes capables de mesurer l'adéquation entre un document visité et les ontologies disponibles, nous sommes alors capables de déterminer le domaine courant de l'utilisateur au fur et à mesure de sa navigation.

En fonction de la navigation de l'utilisateur (pour un utilisateur enregistré avec un profil), nous avons la liste des domaines enregistrés dans son profil et pour les documents visités nous avons, dans certains cas, des annotations qui déterminent le domaine du document. De plus, nous avons la possibilité, avec ce qui a été décrit ci-dessus, de calculer la distance entre un document et un domaine. Enfin, nous conservons dans le profil de l'utilisateur, le domaine courant qui est soit indéterminé (en tout cas à l'initialisation) soit qui a été déterminé auparavant selon les règles que nous donnons ci-après (Figure 24).

Dans le cas où le document est décrit dans un domaine, et si celui-ci est enregistré dans le profil de l'utilisateur, alors nous basculons immédiatement dans ce dernier comme domaine courant de l'utilisateur. Si le domaine n'est pas dans le profil de l'utilisateur, alors ce dernier a le choix de le rajouter dans son profil ou de passer en mode indéterminé.

Dans le cas où le document est décrit dans plusieurs domaines, alors soit l'utilisateur force le domaine courant, soit on détermine le domaine le plus adéquat en fonction de la distance isotopique.

Enfin, pour tous les documents qui n'appartiennent à aucun domaine, nous calculons la distance isotopique avec les domaines enregistrés dans le profil de l'utilisateur et si un score élevé apparaît, nous proposons alors à l'utilisateur de confirmer l'appartenance à tel ou tel domaine.

Document	URL non documenté	URL documenté dans un seul domaine	URL documenté dans plusieurs domaines
Utilisateur			
Utilisateur connu, domaine inconnu	Evaluation de la distance	<ol style="list-style-type: none"> 1. Domaine dans le profil utilisateur: bascule immédiate. 2. Domaine non présent dans le profil: <ol style="list-style-type: none"> a. L'utilisateur accepte la proposition de domaine. b. Reste en domaine indéterminé. 	<ol style="list-style-type: none"> 1. Si aucun domaine dans le profil, reste indéterminé. 2. Un seul domaine dans le profil: bascule immédiate. 3. Plusieurs dans le profil: évaluation de la distance.

²⁷ Moteur d'indexation et de recherche pour un domaine ou un intranet: <http://www.htdig.org/>

Utilisateur connu, domaine connu	Evaluation de la distance	<ol style="list-style-type: none"> 1. Même domaine: continue. 2. Changement pour un domaine dans le profil: bascule. 3. Nouveau domaine: <ol style="list-style-type: none"> a. L'utilisateur accepte ce domaine: bascule. b. L'utilisateur documente ce document dans son domaine courant. c. Bascule en mode indéterminé. 	<ol style="list-style-type: none"> 4. Domaine existe pour le document: continue. 5. Domaine n'existe pas pour le document: <ol style="list-style-type: none"> d. Autre domaine existe: bascule. e. Autres domaines existent: évaluation de la distance. f. Domaines inconnus: L'utilisateur documente dans un domaine: bascule. L'utilisateur enregistre un domaine dans son profil: bascule. Domaine utilisateur indéterminé.
----------------------------------	---------------------------	---	--

Figure 24 : Onto-plexing automatique

2.4 Conclusion

L'évaluation du domaine courant de l'utilisateur nous permet de mettre en application le concept de cache métier qui permet un caching plus nuancé tenant compte des éventuelles disparités en consommation de différentes communautés. De plus, avec l'évaluation automatique du domaine courant, nous pouvons interagir avec l'utilisateur pour l'accompagner dans la rédaction d'annotations et augmenter ainsi la qualité de celles-ci.

3 Proxy métier: le cas SIRANAU

SIRANAU est un projet mené par le laboratoire de bases de données pour le compte de la Radio Suisse Romande (RSR). SIRANAU signifie: Système Intégré Radiophonique pour les Archives Numériques Audio. Il s'agissait de mettre en place un système d'archivage numérique pour les sons de la radio. Cette dernière produit quotidiennement des émissions, pour la plupart en numérique, mais dispose également de tout un fond d'archives sur différents supports (disques à gravure directe, bandes magnétiques, etc.). Ces archives constituent un patrimoine d'une valeur considérable. Mais la taille du stockage est également conséquente: la RSR compte archiver plusieurs centaines d'heures d'éléments sonores, mais également transférer sur des supports numériques pour les pérenniser des documents actuellement stockés sur des supports analogiques déjà fortement dégradés et souvent déjà dans un état de dégradation irrémédiable. Il s'agissait donc bien de stocker des milliers voire des dizaines de milliers d'heures. Ce projet devait non seulement aborder la question de l'acquisition, du stockage et de la documentation de ces archives, mais également la distribution pour les utilisateurs de la radio qui sont les premiers consommateurs de ces archives. Cette distribution devait présenter les meilleures performances possibles, d'où l'intérêt de mettre en place des proxys pour accélérer les transmissions.

3.1 Contexte

La RSR dispose d'un intranet complexe constitué de plusieurs sous-réseaux bien distincts en partie pour des questions de sécurité, d'autre part aussi pour des questions logistiques

(locaux à Genève et à Lausanne). De plus, cet intranet local s'inscrit dans un réseau à l'échelle nationale qui relie les différentes régions linguistiques.

Enfin, si nous sommes habitués à trouver sur Internet des sons compressés de bonne qualité et de taille raisonnable, les exigences professionnelles d'une radio multiplient la taille des objets par un facteur considérable. Cette taille des objets pose immédiatement des problèmes de distribution et de rapidité d'accès pour les usagers internes à la radio (Se référer à [24] pour plus détails sur la distribution en taille et en nombre des sons).

Cependant, l'application utilise le protocole HTTP pour la consultation et l'acquisition des sons. Par conséquent, l'utilisation de proxys standard est parfaitement envisageable.

3.2 *Méta-données*

Les données décrivant ces sons sont relativement simples et ne constituent qu'un ensemble d'informations documentaires gérées par les archivistes (type d'enregistrement, auteurs, date, intervenant(s), etc.). Ces données existent déjà sous forme de fiches (cartons ! ou dans une base de données documentaires). Deux tables suffisent quasiment au schéma. Quelques segments sont également documentés individuellement pour les cas des interviews longues.

La consultation des données documentaires ne pose par conséquent aucun problème quant à la distribution dans l'intranet de ces informations.

3.3 *Distribution des sons*

Il en va tout autrement pour les sons ! Ceux-ci peuvent facilement dépasser le giga-bytes en qualité professionnelle et le journalisme étant un monde rapide, la consultation des archives ne peut pas souffrir de délais trop importants. D'autre part, les fiches documentaires ne sont pas suffisantes pour pouvoir sélectionner tel ou tel son. Avec la mise en place du service, il allait donc survenir de sérieux problèmes d'encombrement sur l'intranet de la radio.

Le système doit impérativement proposer un stockage central de plusieurs tera-bytes de données gérées par le service des archivistes qui garantit aussi la pérennité des données.

Par contre, on peut imaginer que ces sons soient ensuite dupliqués pour être "rapprochés" des utilisateurs en fonction des besoins.

Il existe deux types de besoins qui peuvent conditionner les recherches des utilisateurs. Premièrement le service, en effet la RSR propose quatre programmes différents, en plus d'un service d'actualité. Deuxièmement, l'actualité et les événements qui surviennent.

Enfin, les sons peuvent être consultés par morceaux choisis. Il est donc intéressant de reprendre ici la réflexion qui a été faite au chapitre 5 sur la composition des documents avec la possibilité de cacher des parties de ressources.

3.4 *Utilisation des proxys pour SIRANAU*

Pour l'utilisation conditionnée par le service, on retrouve le schéma plésiocentrique. L'utilisation des proxys standard répond donc immédiatement à ce problème. En installant un proxy dans chaque service avec un cache suffisant, on est sûr de mettre à disposition très rapidement les sons fréquemment utilisés par les utilisateurs de ce service.

En revanche, pour mettre à disposition des archives en relation avec les événements survenus, le fonctionnement standard du proxy n'est pas suffisant. Pourtant, le nom d'un personnage décédé, un accident en relation avec telle ou telle technologie, ou un incident dans telle ou telle zone géographique, définissent un domaine. En déterminant les mots clés associés à ce domaine, on peut rapidement retrouver toutes les documentations en

relation et donc les sons associés. Ici les fiches documentaires des sons peuvent être traitées comme des annotations de ceux-ci. Il est donc possible de déterminer un ensemble de sons parmi lesquels il est plus que probable que des consultations vont être faites. En provoquant le chargement de ces sons dans le proxy on atteindra notre double objectif: accélérer la vitesse apparente de consultation et diminuer le trafic dans l'ensemble du réseau en "rapprochant" les sons concernés du groupe d'utilisateur concernés.

4 Conclusion

Nous espérons, au travers des analyses et des propositions faites dans ce chapitre, avoir démontré que les proxys ont effectivement un rôle prépondérant à jouer dans le cadre du Web sémantique et les que les bénéfices à en retirer sont certains.

Nous n'avons fait ici qu'évoquer quelques possibilités, mais il va de soit que l'enrichissement apporté par la mise en évidence de la sémantique dans l'information distribuée sur Internet apportera encore d'autres applications possibles, notamment dans le domaine du caching et des proxys.

Nous allons dans le chapitre suivant présenter une architecture qui permet entre autres de mettre en place les propositions faites ici.

LA PLATEFORME I3

Proposition d'une architecture pour la mise en place des concepts proposés et l'extension des proxys vers de nouveaux services

1 Introduction

Dans les chapitres précédents, nous avons proposé un ensemble d'extensions possibles pour les proxys : amélioration du cache, engnose, sessions, etc. Dans cette partie, nous allons examiner l'opportunité de réunir ces différents concepts dans une plateforme: I3 pour Intelligent Interactive Intermediaries. Cette plateforme doit donc supporter tous les concepts qui précèdent dans le cadre d'une application proxy Internet.

Les objectifs pour la mise en place de cette plateforme sont:

- Permettre la mise en place des concepts présentés dans ce travail
- Une architecture flexible qui permette la mise en place de nouveaux services dans un proxy
- Le maintien des propriétés fondamentales des proxys, à savoir la transparence d'utilisation et un comportement passif/réactif (par opposition à proactif dans le cadre des CDN)
- L'utilisation de protocoles et de paradigmes existants

De plus, nous souhaitons bénéficier, suivant les cas, des propriétés suivantes :

- Un déploiement "gratuit" (donc pas ou un minimum de modifications pour le client ou le serveur)
- Le caractère engnostique

Pour cela, nous allons commencer par examiner quelques applications existantes, dans le domaine des proxys et des outils d'annotation, qui répondent en partie aux objectifs que nous nous sommes fixés. Pour chaque cas, nous examinerons dans quelle mesure elle satisfait à nos besoins et quelles sont ses faiblesses dans le contexte que nous ciblons.

Nous donnerons ensuite les principes généraux pour la mise en place d'une application conforme à nos objectifs. Pour cela, nous introduisons le concept de proxlet : un agent qui puisse être chargé dynamiquement dans le proxy pour y ajouter de nouveaux services. Malgré l'enrichissement en fonctionnalités du proxy, le caching reste un service central. C'est pourquoi nous proposons d'étendre la modélisation du cache afin de bénéficier des dimensions acquises au niveau de la perception passive de notre proxy.

Sans rentrer dans tous les détails, nous examinerons quelques points marquants de notre plateforme au niveau de l'implémentation. Nous détaillerons notamment la base sur laquelle nous avons choisi de construire la plateforme I3, et quelques points particuliers comme les sessions, alors que nous ne disposons pas des proxy-cookies et de la possibilité d'interagir avec l'utilisateur pour le contrôle des services complémentaires.

Enfin, nous concluons par un catalogue d'applications rendues possibles par notre architecture. Dans l'intention de valider la pertinence de notre démarche, nous avons choisi des exemples qui constituent soit des innovations en terme de services, soit des améliorations par rapport à des solutions existantes, soit encore qui s'insèrent de manière élégante dans notre plateforme. Nous avons divisé ces exemples en 6 catégories: filtrage, aide à la navigation, business intelligence, mobilité, intégration de services et cache amélioré.

Au terme de ce chapitre, nous aurons démontré que les concepts présentés dans les parties précédentes de ce travail : sessions, engnose, sont réalistes puisqu'ils peuvent être implémentés relativement facilement et qu'ils débouchent sur des applications concrètes et d'une utilité certaine.

2 Architecture

Pour notre architecture, nous allons d'abord spécifier les opérations que nous souhaitons supporter. En exposant les opérations que nous souhaitons mettre en place, nous remarquons la nécessité d'introduire un composant : le proxlet, que nous définissons et dont nous donnons les principales caractéristiques. Nous détaillons ensuite les différents composants qui constituent notre architecture : proxlet, gestionnaire de session, serveur d'annotations ainsi que la gestion et le stockage pour le caching.

2.1 Opérations

Au niveau des opérations, notre application doit d'abord se comporter comme un proxy standard comme présenté (voir Figure 4, chapitre 2). Par conséquent, les modifications que nous pouvons apporter se situent sur le flux de requêtes en transit du client vers le serveur et le flux de documents en transit du serveur vers le client (Figure 25, losanges noirs).

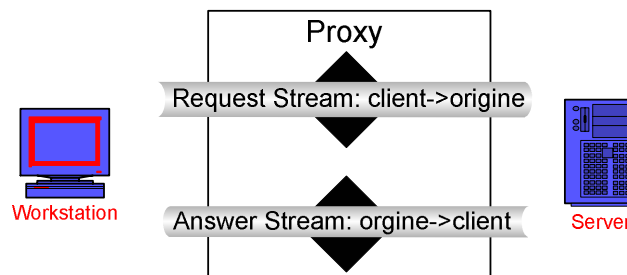


Figure 25 : Intervention sur les processus

Il nous manque cependant à ce stade le moyen d'associer une requête avec une réponse, ce qui nous donne un traitement non coordonné pour une même transaction. Les sessions, qui associent un ensemble de transactions à un même utilisateur, résolvent ce problème et permettent d'établir un lien entre les transactions pour un même utilisateur.

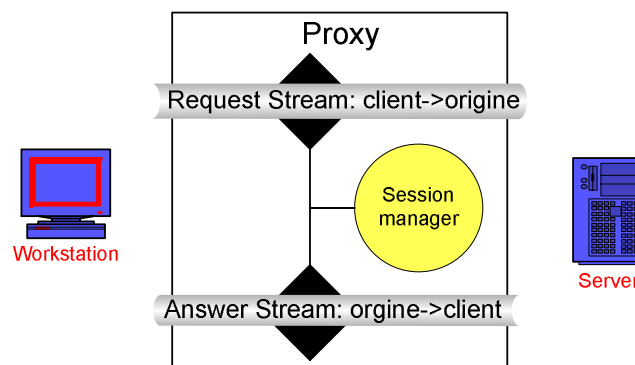


Figure 26 : Processus de gestion de sessions

Afin de pouvoir interagir dans certains cas avec l'utilisateur, il nous faut encore la possibilité de pouvoir générer des documents qui permettent une telle interaction.

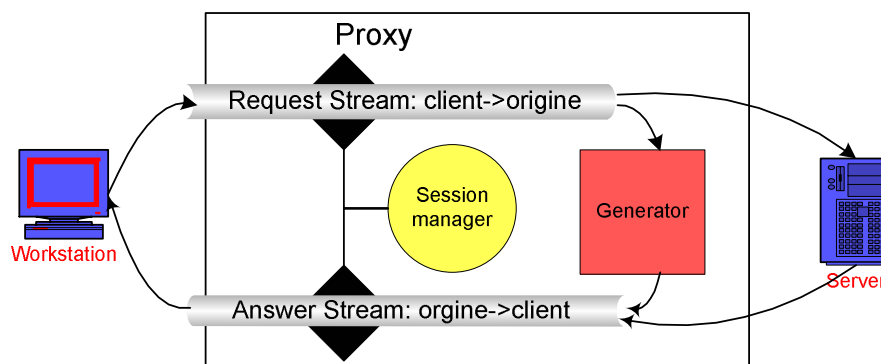


Figure 27 : Agent générateur

2.2 Proxlet

Comme nous l'avons présenté au chapitre précédent, nous souhaitons permettre à une communauté de partager des connaissances. Dans la même optique, il est tout naturel de proposer une plateforme qui puisse s'adapter aux besoins en terme de fonctionnalités pour la même communauté.

Dans cette optique, notre plateforme devra être ouverte au chargement de nouveaux agents en fonction des besoins. A cet effet, nous proposons exactement à l'identique des applets, des servlets et des aglets²⁸: les proxlets. Ces unités constituant des plugins qui, moyennant la définition d'un API standard, peuvent être chargées en temps réel dans le proxy pour y ajouter de nouvelles fonctionnalités. Les proxlets pourront donc intervenir soit comme éditeur de requêtes, soit comme éditeur de réponses (éditeur de ressources). Dans certains cas, afin de permettre l'interaction directe entre l'utilisateur et le proxy à des fins de configuration par exemple, les proxlets pourront également se comporter exactement comme des servlets et produire des ressources envoyées à l'utilisateur.

2.3 Composants

Au niveau de la composition de notre architecture (Figure 28), nous avons d'abord les éléments standard d'un proxy : l'interface et le stockage pour le cache. L'interface proxy supporte le protocole HTTP et implémente les fonctionnalités client et serveur pour exécuter les opérations de : prise en charge des requêtes des clients, récupération du document dans le cache ou relais de la requête vers le serveur d'origine et récupération de celui-ci, et enfin renvoi du document au client en réponse à sa requête.

Au niveau du stockage, étant donné que nous souhaitons détacher l'indexation de la présence ou non des documents dans le cache (chapitre 3), étendre la perception du proxy par l'introduction dans l'indexation de paramètres supplémentaires (chapitre 3), ajouter des propriétés de plus haut niveau (engnose, chapitre 4), nous introduisons un composant SGBD. Ce composant nous permet de mettre en place un schéma plus riche (voir plus loin) qu'un simple enregistrement par document. Nous pourrions ainsi

²⁸ <http://www.aglets.org>

modéliser les liens entre les documents, l'emplacement du document et des informations complémentaires comme les annotations.

Pour le support de l'extension des fonctionnalités, nous mettons en place un environnement d'exécution pour les proxlets. Cet environnement doit être capable de charger en temps réel des proxlets, de fournir les interfaces pour accéder aux autres composants de la plateforme. Il doit également gérer l'exécution des proxlets en établissant une chaîne ordonnée de ces derniers. En fonction des types de proxlets (édition de la requête, de la réponse ou génération) et des conditions d'exécutions (par exemple si le proxy s'applique à tel ou tel type de ressource), il les inclura dans la chaîne d'exécution. A l'exécution, chaque proxlet recevra en entrée respectivement la requête ou la réponse et produira en sortie respectivement une requête ou une réponse traitée.

Nous établissons un proxlet particulier toujours présent : le gestionnaire de sessions qui supporte les spécifications que nous avons présentées au chapitre 4.

Pour la mise en place de l'engnose définie au chapitre 5, nous incluons dans notre plateforme un serveur d'annotations et un gestionnaire de cache étendu qui prenne en compte les informations supplémentaires ainsi mises à disposition et propose une gestion par cache métier comme définie au chapitre 6.

Enfin, pour supporter l'enrichissement du modèle de notre cache induit par l'augmentation des dimensions que nous prenons en compte, nous introduisons une base de données. Nous détaillerons plus loin le nouveau modèle qui, en résumé, vise à stocker les informations sur les ressources séparément de ces dernières et surtout permet de modéliser les différents concepts que nous avons introduits dans les chapitres précédents.

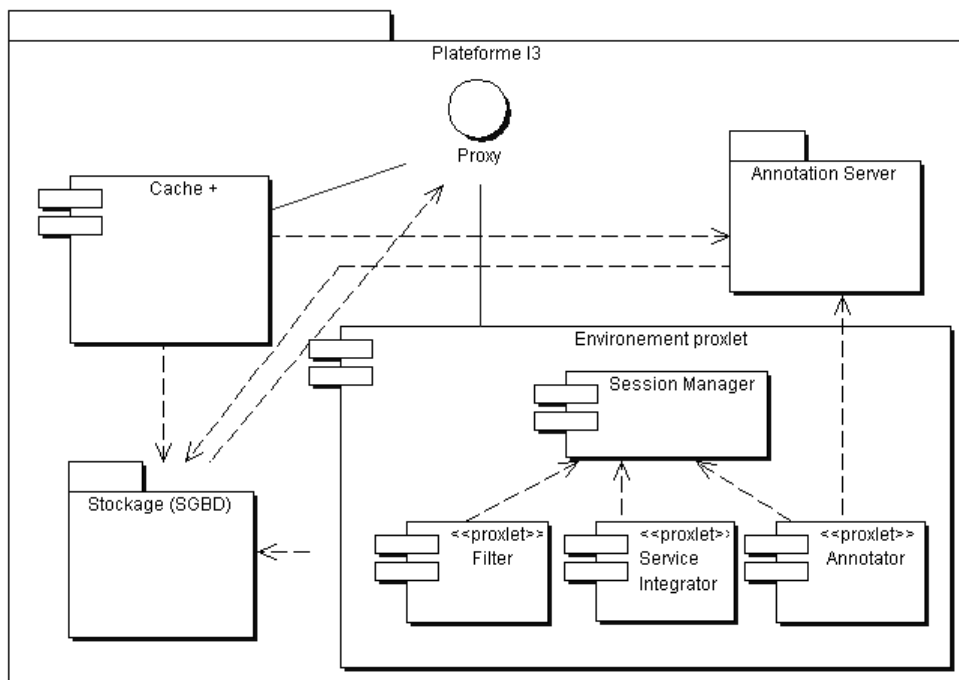


Figure 28 : I3, schéma des composants

2.4 Java

Au vu de ce qui précède, nous avons donc besoin, au niveau de l'environnement du langage, de code mobile et donc d'un maximum d'indépendance au niveau de l'environnement physique et du système et de fonctionnalités orientées vers les applications Web.

C'est pourquoi nous avons choisi de nous concentrer sur l'environnement Java qui nous semble proposer le plus de paradigmes répondant à nos objectifs.

3 Projets similaires

Avant de présenter les détails de notre implémentation, nous donnons ici un bref aperçu non exhaustif des différentes applications que nous avons analysées ou testées et qui nous ont servi soit de base pour la construction de notre application, soit d'exemple pour quelques unes des idées originales qu'elles proposaient.

Nous avons choisi deux types d'applications qui puissent servir de base de réflexion à la réalisation de notre plateforme. Tout d'abord des proxys, puisque c'est le cœur et le point de départ pour la mise en place de nos extensions. Plus spécifiquement, nous avons choisi des applications proposant une certaine flexibilité, un paradigme fortement orienté objet et donc plus ouvert et propre à supporter de nouvelles extensions. Ces applications ont également toutes été développées en Java, puisque nous avons choisi cet environnement.

Dans la deuxième catégorie d'applications présentées, nous examinons des outils et plus particulièrement des serveurs d'annotations, puisque c'est un sous-système dans notre architecture.

3.1 Proxys adaptables

Nous regroupons ici les applications qui proposent des fonctionnalités de caching proxy et intègrent une certaine flexibilité parfois simplement parce que développées en Java.

3.1.1 *Jigsaw*

Jigsaw²⁹ est un projet du W3C qui propose une plateforme de serveur Web écrite entièrement en Java. Jigsaw propose une implémentation du protocole http 1.1. Son architecture est relativement modulaire et permet donc d'ajouter et surtout de configurer facilement au moyen d'un GUI d'administration, des modules apportant telle ou telle fonctionnalité. Un module proxy est proposé parmi d'autres.

3.1.1.1 *Avantages, défauts*

Jigsaw, qui se veut une plateforme exemple pour le W3C, propose un design propre, très académique mais un peu lourd. Les performances s'en ressentent surtout en mode proxy. Dans cette cathédrale, il devient difficile de modifier/ajouter des fonctionnalités aussi spécifiques que celles dont nous avons besoin. En effet, pour avoir testé en redirigeant le stockage des objets dans une base de données, l'intervention est difficile, se répand rapidement en divers endroits du code et le résultat n'est pas concluant en terme de vitesse et d'efficacité.

3.1.2 *MOWS*

De même, Mows³⁰ est à la fois un proxy et un serveur Web distribué écrit en Java. Il est construit sur la base de modules qui peuvent être chargés aussi bien localement que sur un serveur distant (notion de plugin). Il existe quinze types de modules différents et ces types peuvent être étendus pour ajouter de nouvelles fonctionnalités.

²⁹ <http://www.w3.org/Jigsaw/>

³⁰ <http://mows.rz.uni-mannheim.de/mows/>

3.1.2.1 *Avantages, défauts*

Mows est un projet intéressant pour ses possibilités d'extensions. Mais le projet semble stagner depuis 1997 (dernière release). De plus, la catégorisation des modules est complexe et mal modélisée.

Enfin, pour avoir réalisé quelques modifications sur le code à des fins de test, le programme s'est révélé plein de bugs et l'implémentation parfois hasardeuse.

3.1.3 *Muffin*

Muffin³¹ est un projet de thèse développé à l'université de San Diego qui propose essentiellement un système de filtrage. Ce proxy ne supporte pas les sessions, il n'est donc configurable que d'une seule façon, ce qui présuppose une installation et une utilisation individuelle. Les buts de Muffin sont: la sécurité, la protection de la sphère privée, la possibilité d'enlever des fonctionnalités indésirables ou d'ajouter de nouvelles fonctionnalités.

On trouve les exemples suivants parmi les filtres proposés avec la distribution de base:

Animation Killer

Supprime les images GIF animées ou encore les modifie pour qu'elles ne tournent pas en boucle

CookieMonster

Supprime les cookies

Decaf ou NoCode

Supprime tout ce qui est javascript et Java (ce genre de contrôle peut pourtant être fait directement dans la plupart des navigateurs)

DocumentInfo

Affiche des informations supplémentaires sur les documents (dernière modification, taille, etc.)

ForwardedFor

Implémente la directive http: X-FORWARDED-FOR (qui intriduit dans les en-têtes http des informations concernant les agents intermédiaires traversés)

Glossary

Pour tous les mots d'un glossaire (vocabulaire) qui apparaissent dans les documents, introduit une référence vers un lien défini.

History (ré-invente la roue , puisque cette fonctionnalité existe déjà dans les navigateurs)

ImageKill

Supprime les images présentant certaines caractéristiques. Devrait permettre de filtrer les images publicitaires comme les banners qui ont des tailles caractéristiques.

NoThanks

Permet de bloquer l'accès ou de supprimer des éléments de document sur la base d'un système de pattern matching.

Preview

Permet de pré-visualiser dans une fenêtre le document avant qu'il soit envoyé au navigateur.

Referer

Supprime la directive REFERER des en-têtes http.

Rewrite

Réécrit les URL des requêtes pour par exemple forcer des redirections.

Secretary

Complète les formulaires automatiquement avec des valeurs pré-configurées (ce que l'on trouve dans Internet explorer, mais sans la capacité d'apprentissage).

³¹ <http://muffin.doit.org/>

3.1.3.1 Avantages, défauts

Nous n'avons pas effectué de tests particuliers avec Muffin notamment en termes de performances. L'approche est intéressante, elle permet d'introduire de nouveaux filtres relativement facilement et le code est compact. De plus, de nombreux filtres sont fournis comme exemples et démontrent l'utilité d'un tel système.

En revanche, il n'y a pas de support pour les sessions, et aucun API ne permet de le faire simplement.

3.1.4 IBM-WBI

WBI³² signifie: Web Intermediaries. Il peut être utilisé aussi bien comme proxy que comme serveur Web. Les intermédiaires sont définis comme des unités de traitement qui peuvent être placées tout au long du flux d'informations. Ces unités sont programmées pour façonner, personnaliser ou encore compléter les données en transit. La programmation par intermédiaires est particulièrement intéressante lorsque l'on souhaite conserver le client et le serveur. La fonctionnalité de caching proxy est un exemple d'intermédiaire.

WBI est une plateforme pour créer des applications Web basées sur le concept d'intermédiaire.

Le package WBI propose un certain nombre d'exemples. Dans les fonctionnalités proxy, on retrouve sensiblement le même type d'exemples que dans Muffin.

3.1.4.1 Architecture

Dans ce paragraphe, nous allons aborder le fonctionnement de WBI en décrivant d'abord le processus et les activités de traitement du système, ensuite nous décrivons les différents composants qui entrent en jeu.

WBI est un processeur http : il reçoit une requête http d'un client (navigateur) et produit une réponse http qui lui est renvoyée. Entre deux, le système construit une chaîne de traitements qui effectuent un certain nombre d'opérations. Ces opérations sont regroupées selon trois catégories: l'édition de la requête, la génération d'une réponse et enfin l'édition de la réponse.

Dans chaque catégorie, les opérations sont effectuées par des unités (les intermédiaires) appelées MEG. Il existe donc pour les trois catégories d'opérations, trois type de MEG: RequestEditor, Generator et Editor qui sont disposés dans leurs blocs respectifs.

La chaîne de traitement est construite dynamiquement, pour chaque requête adressée au système qui détermine cette chaîne en fonction de règles d'activation.

Chaque MEG instancie ses règles d'activation. Lorsque WBI reçoit une requête, il instancie et enchaîne alors tous les MEGS dont les règles d'activations retournent vrai.

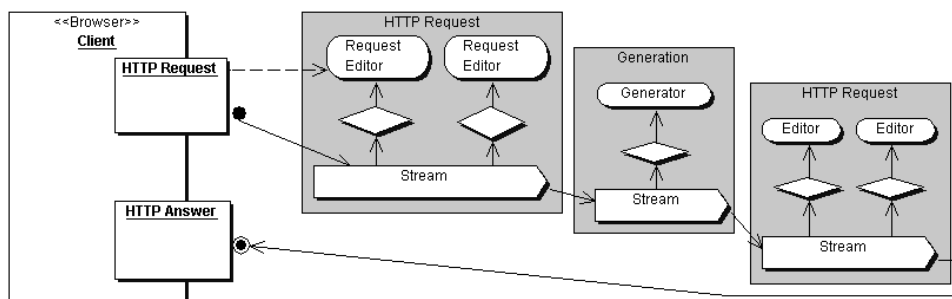


Figure 29 : Chaîne de traitement des requêtes dans WBI

³² <http://www.almaden.ibm.com/cs/wbi/>

Au niveau des composants, nous retrouvons comme élément de base les MEGs qui sont regroupés dans des unités d'installation que sont les Plugins. Un plugin constitue également un environnement d'exécution partagé par les MEGs.

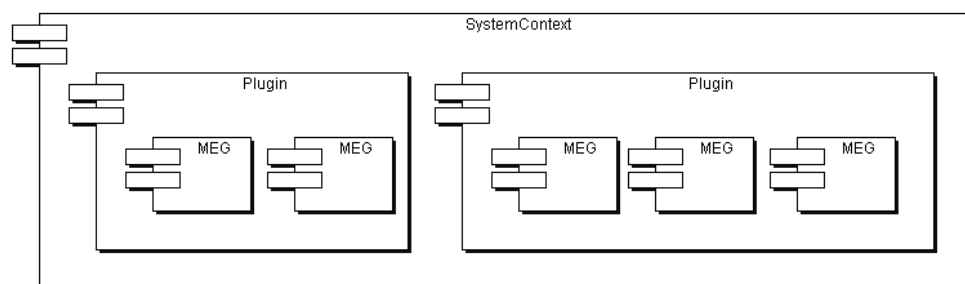


Figure 30 : Composants WBI

Pour créer une application, il faut donc implémenter un certain nombre de MEGs qui effectuent le traitement souhaité : modification de la requête, génération du contenu et modification du résultat. Ces MEGs sont regroupés dans un plugin qui peut être installé dans le serveur WBI et qui instancie les MEGs.

3.1.4.2 Avantages, défauts

L'avantage de WBI est un API extrêmement bien conçu, une très grande flexibilité dans la possibilité d'introduire des modules tout en conservant une relative simplicité dans le design de nouvelles applications. Le niveau d'abstraction est suffisamment général pour que le champ d'applications possibles soit bien plus large que dans les produits précédents.

De plus, même avec un certain nombre de modules (plugin) introduits, la vitesse d'exécution est encore plus qu'acceptable.

Le seul défaut est que ce soit un produit commercial, distribué librement pour le moment (alpha work d'IBM), mais sans les sources. En revanche, la documentation pour construire des plugins est bien faite et il est donc très facile de construire de nouvelles applications sur cette base.

Nous avons choisi cet outil pour la validation de nos travaux.

3.1.5 Synthèse

Une première catégorie apparaît dans les projets présentés : celle des serveurs Web qui, en bonus, proposent la fonctionnalité de proxy (MOWS et JigSaw). Comme le couteau suisse intégrant quarante outils, ils sont compliqués à utiliser, lourds, peu commodes et surtout manquent de performance. La fonctionnalité proxy est déjà une extension des fonctionnalités existantes, et il apparaît peu commode d'inclure des fonctionnalités supplémentaires dans ce qui constitue déjà une extension.

Une autre catégorie est composée des outils qui sont destinés à être utilisés comme des proxys individuels. Muffin en est un exemple, mais il en existe une pléthore que nous n'avons pas détaillée ici. Ces outils "égoïstes" sont configurables par l'utilisateur, fonctionnalité que nous souhaitons proposer, mais ne permettant aucune mise en commun de ressources.

Enfin, WBI est une solution déjà plus intermédiaire. Il peut être utilisé comme serveur Web ou comme proxy, mais la fonction proxy n'est pas une simple extension. Le proxy s'intègre complètement dans l'architecture générale de l'application grâce au découpage généraliste : éditeur de requêtes, générateur, éditeur. Par contre, le mode proxy a été

conçu également pour être utilisé en mode individuel. Il n'y a pas de fonctionnalités pour un support multi-utilisateurs.

Enfin WBI correspond exactement, dans sa conception, à l'architecture que nous souhaitons mettre en place.

3.2 Outils d'annotation

Les projets d'annotation pour le Web reprennent l'intention initiale de Tim Berners-Lee. Nous ne proposons ici que d'examiner deux des solutions que l'on trouve parmi de nombreuses autres sur le Web. La première est une solution open source qui s'intègre dans le projet Mozilla. La deuxième fait partie d'un programme plus large : le projet OntoAgents conduit à l'université de Stanford. Ces deux exemples représentent deux approches très différentes avec, d'un côté, une solution simple permettant juste de typer les annotations, et de l'autre, une solution qui vient s'intégrer dans un projet beaucoup plus large incluant la notion d'ontologie.

3.2.1 *Annozilla*

Annozilla³³ a l'avantage de s'intégrer complètement dans le navigateur Mozilla. Son utilisation est donc parfaitement transparente pour l'utilisateur. Son usage est également très simple : l'utilisateur a la possibilité de "coller" des annotations où il le souhaite sur les documents qu'il visite. Il peut éventuellement filtrer les annotations par langue et celles-ci ont un type: explication, commentaire, etc.

Annozilla est une application client qui se connecte sur un serveur Annotea, il en existe un, accessible à tout le monde, en démonstration au W3C.

Annotea³⁴ est un serveur qui permet de stocker des annotations qui font référence à n'importe quel document ou partie de document (grâce à XPointer, voir chapitre précédent). C'est un projet du W3C composé de cgi-script Perl qui doivent être installés avec un serveur Web et stockent les annotations dans une base de données MySQL selon un schéma RDF³⁵.

Ce projet fait partie du développement du Web Sémantique pour le W3C.

Une instance de ce serveur est en accès public au W3C. Cette démonstration apporte par contre très rapidement, la preuve qu'un service d'annotation ouvert à toute la communauté Internet pose un réel problème de prolifération. Il suffit de se rendre sur la page d'Annozilla pour découvrir une profusion d'annotations dénuées de sens qui ont été déposées en partie par erreur, par des utilisateurs voulant simplement faire des tests.

3.2.2 *OntoMat-Annotizer*

OntoMat³⁶ annotizer est un outil qui s'inscrit dans le projet Onto-Agent³⁷, dont le but est la mise en place d'une plateforme pour des agents qui consomment de l'information issue des efforts du Web sémantique (ce qui s'inscrit pleinement dans la définition "qui peut être traité par une machine" donnée par Tim Berners-Lee). La base du projet est l'exploitation d'une ontologie définie comme: "la spécification explicite d'une conceptualisation qui fournit toute la terminologie requise, et une base d'échange pour une communauté partageant les mêmes intérêts". Cette plateforme inclut donc de facto

³³ <http://annozilla.mozdev.org/>

³⁴ <http://www.w3.org/2001/Annotea/>

³⁵ <http://www.w3.org/RDF/>

<http://www-db.stanford.edu/~melnik/rdf/api.html>

³⁶ <http://annotation.semanticweb.org/ontomat.html>

³⁷ <http://www-db.stanford.edu/OntoAgents/>

les ontologies comme base pour la structuration des annotations. On retrouve ainsi dans [73]: "... une plateforme d'annotation qui permet la construction de méta-données relationnelles i.e. : des méta-données qui sont constituées d'instances de classes et d'instances de relations. Ces instances ne sont pas basées sur une structure fixe, mais sur un domaine ontologique".

3.2.3 Synthèse

Pour l'utilisateur, la configuration d'un serveur d'annotations est très similaire à celle d'un proxy. En ajoutant les fonctionnalités d'annotation dans le proxy, nous rendrions l'utilisation de cette fonctionnalité plus accessible.

Pour l'administrateur, toute la couche de communication nécessaire à la mise en place d'un serveur d'annotations est déjà en place dans un proxy. Le stockage dynamique d'informations est déjà partiellement en place, d'autant plus si nous enrichissons les possibilités de stockage du cache comme nous l'avons suggéré.

Comme nous l'avons déjà démontré au chapitre précédent, en plus du gain en transparence et en facilité d'utilisation, en incluant les annotations dans le proxy, nous augmentons les informations qui peuvent être prises en compte pour la gestion du cache et la mise à disposition de nouveaux services (voir plus loin).

Cependant, à l'heure actuelle, les solutions qui fleurissent apportent chacune leurs spécificités et surtout leurs incompatibilités : outils d'annotations basés ou non sur une ontologie, différents format d'annotation, etc. Le succès gigantesque du Web repose en partie sur une normalisation, en tout cas au niveau d'http, extrêmement suivie. On peut peut-être espérer qu'en proposant une solution intégrée au proxy, on puisse rapidement fédérer les efforts.

4 Modélisation du cache

Si l'on souhaite stocker de nouvelles informations dans le cache, notamment pour les annotations et les ontologies, la modélisation du cache doit être adaptée. Ceci est également vrai pour les informations évoquées dans le chapitre 3 avec par exemple le stockage d'informations sur les liens entre les documents. De plus, comme nous l'avons fait remarquer au chapitre 2, le fait de disposer d'un index ne portant que sur les ressources qui se trouvent dans le cache peut conduire à des aberrations. D'autre part, pour les différentes dimensions dont nous souhaitons tenir compte pour la politique de gestion du cache, un index simple (une seule relation), ne suffit plus. Enfin, d'autres éléments liés aux fonctionnalités que nous souhaitons adjoindre à notre proxy, nécessitent également une base de données plutôt qu'un simple index.

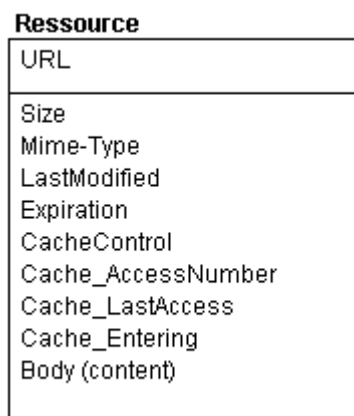


Figure 31 : Modèle "classique"

A partir d'un modèle "classique" de stockage des données dans le cache (Figure 31), nous proposons une modélisation plus complète qui permette de modéliser l'environnement du document et les méta-données que nous souhaitons lui associer.

4.1 Ressource et propriétés

Nous regroupons ici les propriétés intrinsèques du document ainsi que les statistiques d'utilisation. Ces informations sont présentes dans le modèle classique, mais dans notre modèle, nous "détachons" la ressource de ces informations pour deux raisons : nous évitons les aberrations évoquées dans le chapitre 2 (avec un nombre de requêtes égales, un document suscitant une avalanche de demandes dans un laps de temps court l'emporte systématiquement), enfin pour des clones d'une même ressource nous ne stockons qu'une seule copie.

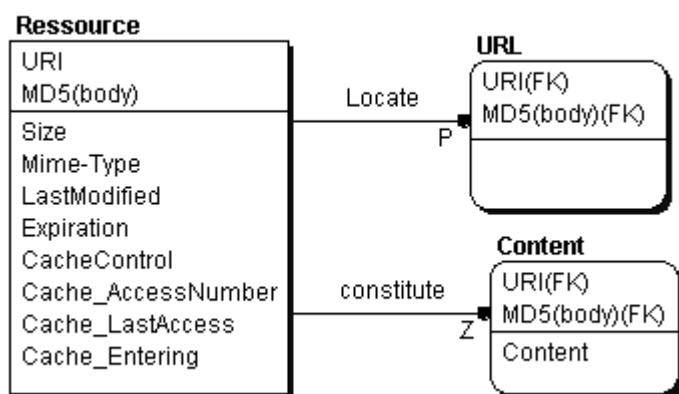


Figure 32 : Ressource et propriétés

Dans ce modèle, nous dissocions tout d'abord le document du descripteur qui peut donc rester dans l'index du cache, même si le document est éliminé par la politique de remplacement. Cette façon de procéder implique bien entendu un remplissage de la base de données au niveau des descripteurs qui s'accumulent indéfiniment. Pour une utilisation à large échelle et à long terme, il faudrait sans doute prévoir une politique de remplacement également pour ces descripteurs. Mais il est de toute façon possible de limiter les aberrations évoquées en permettant à un nombre plus important de descripteurs que de documents d'être conservés. De plus, les descripteurs auront presque systématiquement une taille beaucoup plus faible que les documents.

D'autre part, avec le calcul d'une clé MD5 ou l'utilisation de l'URI pour identifier le document, nous nous donnons la possibilité de ne garder qu'une seule copie du même document dupliqué dans plusieurs endroits distincts.

La clé MD5 d'une ressource est unique et surtout, contrairement aux URL, dépend uniquement des propriétés "physiques" de la ressource. Le gestionnaire du cache a donc la possibilité de détecter des objets clonés stockés à des adresses différentes et de ne stocker qu'une seule copie.

4.2 Ressource et environnement

Par environnement du document nous entendons sa position absolue (le serveur sur lequel il est situé) et sa position relative, c'est-à-dire ses liens sur d'autres documents ou les liens qui le référencent. Nous avons proposé au chapitre 3 une nouvelle politique de gestion du cache qui exploite ces informations ainsi que des procédés de prefetching par

prédictions statistiques basées sur la probabilité des chemins. Nous avons par conséquent besoin de ces informations dans notre modélisation.

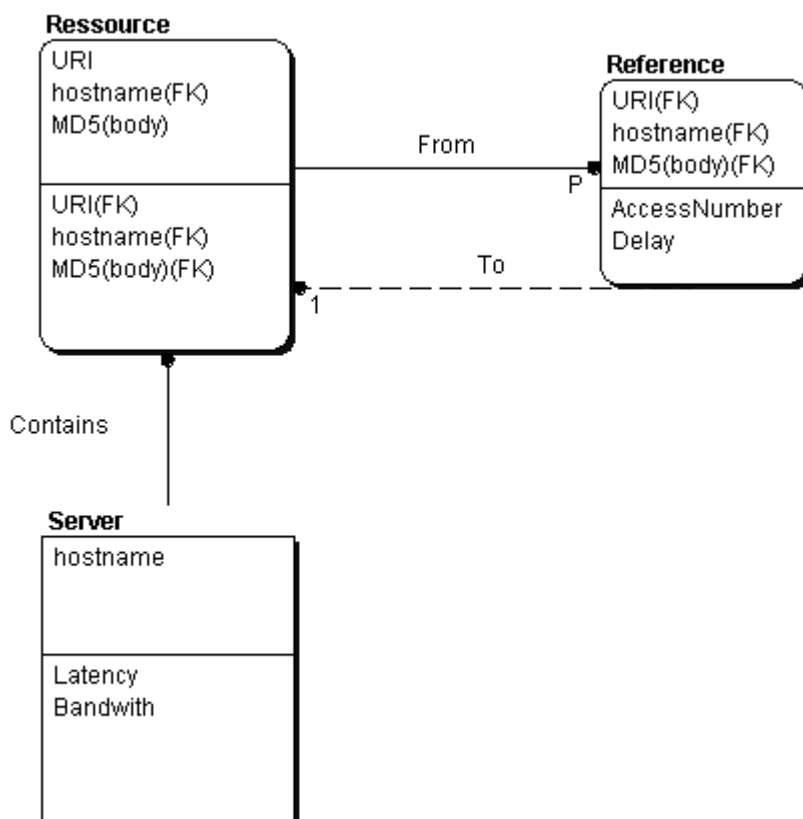


Figure 33 : Ressource et environnement

Comme le montre le schéma (Figure 33), nous modélisons ici la position absolue du document : serveur d'origine et propriétés visibles de celui-ci, ainsi que sa position relative par rapport aux autres documents qu'il référence ou qui le référencent et les statistiques de fréquentation des liens entre les documents.

Sa position absolue est modélisée par le serveur sur lequel se trouve la ressource. Les propriétés du serveur sont la bande passante qui correspond au temps de téléchargement d'un objet en fonction de sa taille et le temps de réponse qui correspond à l'intervalle entre le moment où la requête est envoyée au serveur et l'instant où la réponse commence à arriver.

Au niveau de la modélisation des liens entre les documents, nous stockons les statistiques d'utilisation. Celles-ci regroupent le nombre de fois que ce lien est suivi et l'intervalle de temps moyen que l'utilisateur passe sur la ressource précédente avant de suivre le lien. Sur ce dernier point il est important de noter que ceci ne correspond pas exactement à la durée d'attente que l'utilisateur porte au document. Nous distinguons la durée de stationnement de l'utilisateur en fonction des liens suivis. Si l'on devait définir un temps d'attente par document (qui serait alors une propriété de la ressource elle-même !), il correspondrait à la moyenne de ces moyennes.

Enfin, cette dernière information ne peut bien entendu être "extraite" que si le mécanisme de maintien de session existe. Ce n'est qu'au sein d'une session utilisateur que ces intervalles sont mesurables.

4.3 Informations complémentaires

Nous avons suggéré aux chapitres 4 et 5 l'ajout de nouvelles dimensions par l'extension de la perceptibilité du proxy à l'engnose. Ceci sous-entend que les proxys aient accès à ces informations qui doivent donc également être modélisées.

Les informations complémentaires du document sont les annotations et leur référence à une ontologie.

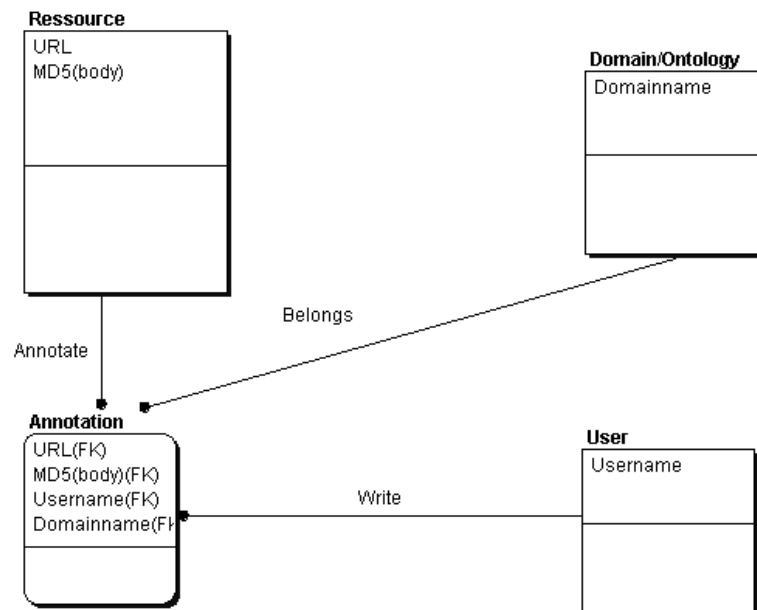


Figure 34 : Annotations et domaine

4.4 Conclusion

Bien entendu, une modélisation plus complète est toujours possible. Des informations comme qui a accédé à ce document, ou encore quand tel utilisateur y a-t-il accédé pour la dernière fois, ne peuvent pas être retrouvées. Nous avons cependant à dessin souhaité rester dans un schéma relativement simple pour ne pas surcharger l'indexation des ressources dans le cache.

Notre modèle est donc minimaliste, mais permet l'implémentation de toutes les fonctionnalités présentées dans ce travail.

5 Implémentation

Nous allons ici détailler quelques éléments de l'implémentation de notre système. Dans un premier temps, nous exposons brièvement une solution simplifiée: TomProxy. Cette solution a l'avantage d'être extrêmement simple: un servlet de quelques dizaine de lignes de Java permettant déjà de valider le fonctionnement de notre système au niveau des sessions et des interactions avec les utilisateurs. De plus, l'API proposé par SUN et supporté par Tomcat englobe énormément de fonctionnalités permettant de développer facilement l'ensemble des fonctionnalités décrites. Nous avons cependant abandonné cette plateforme, d'abord parce que souffrant de performances très mauvaises, ensuite

parce que WBI propose encore un niveau d'abstraction plus élevé et supporte déjà les concepts de plugin et MEG qui correspondent tout à fait à notre sous-système pour les proxlets.

5.1 *TomProxy*

C'est certainement la solution la plus simple: une servlet sur laquelle sont redirigées toutes les requêtes adressées au serveur, et qui elle-même redirige les requêtes sur le serveur d'origine. On transforme ainsi un serveur (en l'occurrence tomcat) en proxy avec une dizaine de lignes de code (sans le caching !). De plus on dispose de tout l'API J2EE avec la gestion des cookies, des sessions, etc.

5.1.1 *Avantages, défauts*

L'avantage est clairement la simplicité et la facilité de mise en œuvre. Le défaut survient cependant rapidement dans les performances qui se sont révélées catastrophiques.

5.1.2 *WBI*

Nous avons finalement, après ces tests, opté pour l'utilisation de la plateforme WBI d'IBM pour construire notre application. Les MEGs et les plugin comme définis dans cette plateforme constituent exactement la même infrastructure que celle définie avec le concept de proxlet.

5.2 *Sessions*

Nous avons proposé au chapitre 4 une implémentation des proxy-cookies pour rendre possible les sessions entre l'utilisateur et le proxy. Comme il n'existe pas d'implémentation de notre proposition et qu'elle serait relativement complexe à mettre en œuvre à seule fin de test, nous proposons ici une implémentation similaire mais simplifiée de notre proposition, qui permet cependant de totalement valider notre démarche.

La différence fondamentale entre les cookies classiques et les proxy-cookies concerne la portée. Pour résoudre ce problème, nous modifions Mozilla pour que dans le cas d'un domaine particulier, il accepte toujours ce domaine, quel que soit le domaine du serveur d'origine, et que d'autre part, il renvoie toujours le cookie, quel que soit le serveur de destination.

Du côté du proxy, nous installons un gestionnaire de cookies qui est capable d'introduire une directive pour l'enregistrement d'un nouveau cookie s'il fait défaut dans la réponse envoyée par le serveur. Le gestionnaire de session fait usage de gestionnaire de cookies pour insérer des cookies qui identifient la session de l'utilisateur et gère une liste de sessions valides et les invalide si nécessaire.

5.3 *Affichage et interaction*

Afin de proposer un service transparent pour l'utilisateur, il est bien entendu indispensable de minimiser les demandes d'interaction. Cependant, déjà pour l'authentification de l'utilisateur afin de pouvoir établir une session avec ce dernier, une telle interaction est inévitable.

A cet effet, nous introduisons, dans la première ressource de type document html demandée, lors de la création de la session dans l'application, une fonction javascript qui commande l'ouverture d'une nouvelle fenêtre avec l'appel à un document local au proxy. Ce document contient l'initialisation de l'interaction entre l'utilisateur et le proxy. C'est la seule transaction qui nécessite absolument la modification d'un document html demandé et constitue donc une violation de la transparence.

Cette fenêtre de contrôle ouverte, on peut initier toutes les interactions avec l'utilisateur en conservant une navigation indépendante et transparente avec l'utilisateur. Cette fenêtre de contrôle permet alors à l'utilisateur de s'authentifier (form html), puis de configurer les services dont il souhaite disposer, ouvrir d'autres fenêtres pour afficher des informations complémentaires ou encore interagir avec des outils comme une application d'annotation.

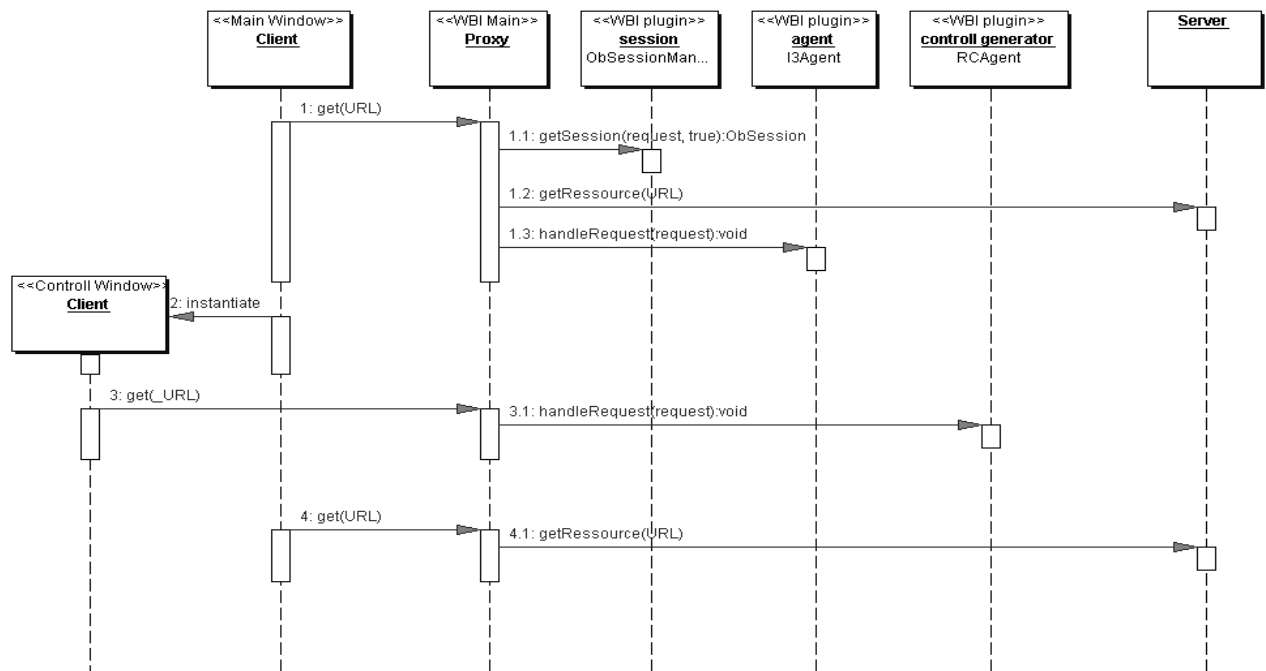


Figure 35 : Séquence d'exécution à l'initialisation

On le voit sur le schéma (Figure 35), lorsqu'un utilisateur initialise une session avec le proxy, le résultat de la première requête est traité par un agent qui introduit une fonction javascript (étape 1.3). Cette fonction induit la création d'une nouvelle fenêtre client (étape 2) qui va alors indépendamment s'adresser au proxy avec des demandes d'URL particuliers qui ne correspondent pas à des adresses existantes (étapes 3, 3.1). A partir de là, le client poursuit sa navigation de manière transparente (étapes 4 et suite).

5.4 Prototype

Une implémentation des différents éléments (session, fenêtre de contrôle, version modifiée de Mozilla) se trouve à télécharger sur la home page du projet I3: <http://lbdpc15.epfl.ch/I3/>.

6 Applications

La plateforme que nous proposons avec les fonctionnalités décrites, permet de mettre en place un nombre d'applications seulement limité par l'imagination. On pourra déjà s'inspirer des solutions disparates qui foisonnent autour des différentes versions de navigateurs. Notons à ce propos que notre solution a l'avantage de permettre de distribuer de tels "goodies" à tout type de navigateur.

Nous proposons ici quelques applications qui illustrent l'intérêt d'une telle plateforme.

6.1 *Filtre*

6.1.1 *Protection parentale*

Il existe déjà pléthore de solutions de filtrage pour la protection des enfants qui naviguent sur Internet. Ces solutions qui s'installent sur l'ordinateur familial sont pourtant facilement contournables (il suffit par exemple d'installer un autre logiciel de navigation !) et gageons que des adolescents auront vite fait de mettre en place une telle solution. Alors que la redirection au niveau IP de toutes les requêtes sur le port http sur un proxy est déjà nettement plus difficile à contourner, cela rend le filtrage au niveau du proxy beaucoup plus sûr.

Le filtrage peut s'effectuer d'abord sur les URL avec une liste noire, mais également par le contenu. Il existe aussi des systèmes de protection qui se basent sur l'apparition de mots clés, mais certains mots ont facilement un sens ambigu et seul le contexte peut permettre d'en déterminer la valeur. En reprenant le scoring défini au chapitre précédent et en définissant une ontologie regroupant les thèmes offensants, on peut ainsi mettre en place un système de censure qui ne soit pas rendu obsolète rapidement par la prolifération du Web.

6.1.2 *Surlignage*

A l'instar de Google qui propose d'afficher les documents d'une recherche en surlignant avec des couleurs des mots-clés, on pourrait très bien demander au proxy de mettre en évidence certains termes ou URL.

6.1.3 *Navigation anonyme*

De nombreux utilisateurs souhaitent garder l'anonymat sur Internet et configurent leur browser pour refuser les cookies. Cependant, certains sites ne fonctionnent pas si les cookies ne sont pas supportés. En supportant les cookies au niveau du proxy, pour la durée d'une session, on permet à l'utilisateur d'avoir accès à toutes les fonctionnalités de tels sites, sans avoir pour autant à accepter les cookies au niveau de son browser.

6.1.4 *Filtrage de la publicité*

La publicité sur Internet est parfois particulièrement envahissante : apparition successive de fenêtres résistantes en premier plan, images en surimpression du texte, etc. Sur le même principe que pour la protection parentale, il est possible de filtrer les URL qui pointent vers des sites de publicité. Mais plus efficace, on pourra également désactiver toutes les fonctions javascript qui font appel à la fonction d'ouverture de nouvelles fenêtres sans avoir à désactiver complètement javascript.

6.2 *Aide à la navigation*

On parle beaucoup des autoroutes de la communication, mais un constat s'impose : le Web ne constitue pas un espace particulièrement bien signalisé ! Il existe des outils comme les moteurs de recherche ou les index qui permettent, parfois, de retrouver ce que l'on cherche. Mais il arrive très souvent que l'on ait visité un document que l'on voudrait retrouver, mais que l'on ne sache plus comment y parvenir. Et souvent aussi, les moteurs de recherche fournissent tellement de milliers de documents que trouver le bon revient à retrouver une aiguille dans une meule de foin.

Les dimensions que nous avons ajoutées dans notre proxy sont autant d'informations pertinentes qui pourraient servir d'aide à la navigation. Tout d'abord, dans l'ordre de présentation d'une recherche, il serait pertinent de mettre en tête de liste les documents annotés, donc ceux auxquels d'autres utilisateurs ont accordé de l'importance. Enfin, l'expression des requêtes par des mots clés uniquement n'est pas toujours satisfaisante. Il faut souvent manuellement essayer plusieurs formulations avec différents synonymes et

parfois le mot clé qui décrit le plus judicieusement le document n'est pas présent dans celui-ci. Pour les documents annotés et classifiés dans une ontologie, il sera par contre possible d'inclure dans la recherche tel ou tel domaine sans avoir à en spécifier les mots clés.

6.2.1 Graphe de navigation

Dans les chapitres précédents, nous avons enrichi les informations que contient le cache aussi bien sur les documents qui traversent le proxy que sur les liens qui les relient. Toutes ces informations sont aussi bien des propriétés intrinsèques des documents qu'extrinsèques et découlant de l'usage fait par la communauté d'utilisateurs. Toutes ces dimensions se révèlent évidemment pertinentes pour une meilleure gestion du cache, mais elles pourraient également être autant d'indications pour faciliter la navigation d'un utilisateur de la communauté.

Premièrement, le fait de disposer d'un graphe des documents (propriété intrinsèque à la toile du web) avec les liens qui les relient constitue déjà une représentation facilement visualisable.

Dans les dimensions intrinsèques au document lui-même, pour tous les documents qui ont été annotés et classifiés, le ou les domaines d'appartenance des documents peuvent également se révéler une information pertinente. En représentant ainsi le graphe environnant le document actuellement visualisé par l'utilisateur avec une codification visuelle pour représenter l'appartenance à des domaines, l'utilisateur pourrait rapidement saisir s'il se trouve dans un environnement déjà largement documenté, homogène, étendu, etc. Si le document appartient à plusieurs domaines, il pourrait de plus très rapidement s'orienter pour se maintenir dans le domaine qui l'intéresse (dans l'exemple plus bas, il est plus facile de s'orienter pour trouver des informations touristiques sur une région donnée et d'écarter les documents à caractère purement géographique).

Dans les dimensions extrinsèques, pour la plupart résultant de l'usage, on pourra également représenter la fréquentation des liens et, puisque l'on dispose de sessions utilisateur, le temps qu'un utilisateur met avant de suivre un lien (distance temporelle).

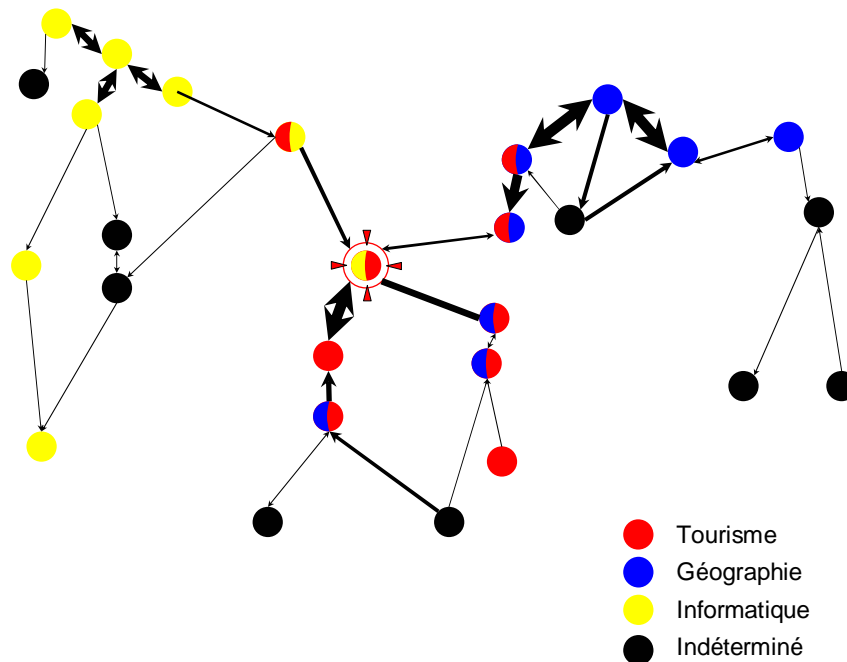


Figure 36 : Exemple de représentation graphique possible pour l'aide à la navigation

Les possibilités de représentation sont infinies. Dans la Figure 36, nous avons par exemple choisi des couleurs pour les domaines décrits, la longueur des arcs pour la distance temporelle, et enfin l'épaisseur du trait pour la fréquentation. De plus, nous n'avons pas représenté les doubles arcs (aller-retour), étant donné la difficulté qu'il y aurait à représenter des fréquentations ou des longueurs dissymétriques.

Nous n'avons pas choisi d'approfondir les possibilités de représentation dans notre travail, mais des techniques comme le Fisheye view [91] permettraient sans doute d'obtenir des résultats encore plus intéressants avec par exemple des regroupements de documents par thème ou par proximité temporelle à la périphérie.

6.2.2 *Visualisation avancée des liens*

Les nouvelles normes XML, comme par exemple XLink posent un certain nombre de problèmes pour les utilisateurs, alors que les navigateurs ne supportent pas à l'heure actuelle de telles fonctionnalités. De plus, ces liens sont beaucoup plus compliqués à afficher dans un document, puisqu'ils ne constituent plus seulement une zone de pointage vers une autre ressource à afficher, mais sont eux-mêmes porteurs d'information (type de lien, commentaires, nom, description, etc.).

Il est plus que probable que les premières implémentations supportant ces liens seront variées et confuses pour l'utilisateur. Comme il est souligné dans [92], il n'existe pas encore de convention pour l'affichage de tels liens.

La mise en place d'un outil dans le proxy permettrait d'une part une généralisation de l'outil de visualisation indépendamment du navigateur tout en le rendant configurable pour l'utilisateur. De plus, cette configuration serait mobile (voir plus loin).

6.3 *Business intelligence*

Internet constitue une mine d'information pour les analystes économiques et financiers : bilans, documents d'analyse du marché, cotations, etc. Ces informations sont le plus souvent bon marché voire gratuites, mais malheureusement coûteuses en temps de recherche. De plus, elles sont souvent disparates, disséminées en différents endroits du Web. Pour reprendre la pensée de Boileau³⁸, nous proposons ici de présenter plus clairement les informations pour donner une vision plus claire à l'utilisateur. Pour cela, nous proposons d'intégrer dynamiquement ces informations "autour" de la ressource que l'utilisateur visite.

6.3.1 *Document rating*

À l'instar de google qui propose cette fonctionnalité³⁹, il serait extrêmement facile d'afficher les statistiques de consultation d'un document, d'autant plus que cette statistique d'utilisation fait partie des paramètres essentiels du caching !

6.3.2 *Outil d'analyse type tableau de bord*

Il existe sur Internet des bases de référence que ce soit pour les cotations⁴⁰, les analyses⁴¹, les bilans⁴², etc. Celles-ci sont relativement bien structurées, mais ne donnent pas toujours l'information recherchée. Il faut alors compléter avec des recherches parfois longues et

³⁸ « Ce qui se conçoit bien, s'énonce clairement. »

³⁹ Google bar: un utilitaire qui s'installe comme un plugin dans le navigateur et propose un certain nombre d'outils: rating, documents parents, documents similaires, etc.

⁴⁰ <http://finance.yahoo.com/>

⁴¹ <http://news.ft.com/surveys>

⁴² <http://www.kompass.com/kinl/index.html>

fastidieuses et surtout, pour obtenir au final un rapport compréhensible, effectuer un travail d'intégration manuel.

Ce que nous proposons ici, c'est un tableau de bord qui afficherait, au fur et à mesure de la navigation, les informations essentielles de base associées au document en cours. Par exemple, l'utilisateur dispose d'une liste de cotations boursières, pour chaque cotation les mots clés associés (qu'il peut redéfinir, compléter). Lorsque le mot clé apparaît dans un document consulté, la cotation boursière apparaît alors dans une fenêtre séparée. Il devient ainsi possible de parcourir les sites de news et de repérer les fluctuations boursières sans avoir à effectuer les recherches et les associations manuellement.

Sans entrer dans l'analyse des besoins et des usages de ce domaine d'activité, nous signalons que d'autres indices, informations servent à l'évaluation dans ce secteur d'activité. L'utilisateur peut donc compléter son "tableau de bord" par association au fil du temps et obtenir une application de surveillance de plus en plus complète et fiable.

De plus, ce genre d'outils peut être étendu à d'autres domaines d'activité :

- Informations géopolitiques en relation avec le pays décrit dans le document ou avec la localisation du document.
- Informations culturelles en rapport avec le film, la pièce de théâtre, l'exposition, etc.
- Informations historiques en relation avec un thème
- Etc.

6.4 *Mobilité*

Notre proposition de proxy permet d'associer une session à un utilisateur et d'y associer un environnement. Par conséquent, notre utilisateur peut retrouver son environnement de navigation instantanément en changeant de poste de travail. Ceci peut se révéler intéressant dans de larges organisations ou dans l'utilisation de "virtual private network" pour des commerciaux ou autres collaborateurs en déplacement. Dans son environnement de travail, on trouve les cookies qui définissent autant de profils sur différents sites, les bookmarks qu'il a collectés, sa configuration proxy, surtout le proxy inclus des services personnalisables et enfin les objets dans le cache à la gestion duquel il a contribué par ses activités.

6.4.1 *Délégation des cookies*

Dès le moment où l'utilisateur peut s'identifier sur le proxy, il peut souhaiter que ses cookies ne soient plus gérés au niveau du navigateur, mais associés à son compte. Ceci lui permettrait de retrouver intégralement et rapidement les différents services dans le même état d'intégration.

6.4.2 *Délégation des bookmarks*

Changement de navigateur, de version, de poste de travail (bureau, privé) sont autant d'occasions pour "perdre" ses bookmarks. C'est parfois fort agaçant ! La délégation des bookmarks sur le proxy apporte une solution simple à ce problème. Dès le moment où l'utilisateur s'est identifié sur le proxy, il retrouve ses bookmarks, où qu'il soit.

6.4.3 *Mobilité Intervache: Roaming Profil*

Un utilisateur en déplacement peut évidemment souhaiter se connecter à son proxy habituel pour retrouver son environnement. Cependant, cette disposition induira presque certainement des contre-performances. Dans cette disposition, on force un routage des requêtes et des réponses sur un nœud probablement externe au réseau local dans lequel l'utilisateur est connecté. On va alors à l'encontre même des avantages en rapidité et en économie de réseau.

Il serait préférable que l'utilisateur puisse, en se connectant au proxy local, bénéficier de son profil en donnant un identifiant universel (simplement par exemple: `username@proxyadresse`) et que son profil soit importé sur le cache local.

6.4.4 *Mobilité et collaboration du cache: Roaming Cache*

Lors d'un déplacement sur un autre proxy, un utilisateur pourrait importer avec son environnement des documents cachés. On propose ainsi un prefetching sur mesure. Cette fonctionnalité pourrait être particulièrement rentable avec la notion de cache métier.

Prenons l'exemple d'un chercheur invité dans une université. Il se connecte sur le proxy local de l'université hôte, s'identifie puis importe son profil utilisateur ainsi que les documents cachés qui concernent son travail. D'une part il profitera directement de ce prefetching, d'autre part, il est probable que dans le cadre du séminaire qu'il donnera, il cite des documents qu'il a lui-même l'habitude de consulter et induise ainsi une nouvelle fréquentation dans l'université hôte.

6.5 *Intégration de services*

Disparate est certainement un qualificatif qui convient parfaitement au Web. On trouve tout ou presque sur la toile, mais pas toujours ce que l'on cherche et rarement dans la forme souhaitée. Partant du fait que nous avons une communauté d'utilisateurs identifiée et un proxy qu'ils partagent, nous proposerons ici un certain nombre de services qui pourront être mis à disposition de cette communauté en intégrant des informations, des documents ou même plus loin des applications distribuées sur Internet.

Plus généralement, nous présentons aussi ici les différentes applications qui pourraient être intégrées au cache pour apporter de nouveaux services aux utilisateurs.

6.5.1 *Annotations*

Nous avons déjà parlé d'intégrer les annotations au proxy dans le chapitre précédent. Au-delà du simple outil d'annotation et de classification des documents rencontrés, ainsi que de l'aide à la navigation, le fait de disposer des annotations permet également de faire des recherches parmi les documents annotés et donc parmi une collection de documents qui ont été retenus pour leur valeur, leur pertinence, par des utilisateurs travaillant dans le domaine concerné.

De plus, au lieu des résultats de recherche qui donnent des extraits entourant les mots clés critères (c.f. Google, Altavista, etc.), l'affichage des résultats de recherche permettra d'une part la mise en évidence du domaine reconnu pour le ou les documents listés, mais également un commentaire bien plus représentatif que l'extrait automatique.

6.5.2 *Partage des bookmarks*

Pour un projet, pour une équipe, le fait de disposer d'une webo-graphie (bibliographie d'URL) peut se révéler un instrument précieux.

Les utilisateurs pourraient en effet très bien partager les bookmarks qu'ils stockent sur le proxy. Les proxys deviendraient ainsi un excellent moteur de synergie. De plus, les annotations mises en place peuvent également servir de description pour ces bookmarks, transformant ainsi le traditionnel index en un catalogue descriptif.

6.5.3 *Search focus*

Dans l'utilisation de moteurs de recherche, il arrive souvent que l'on trouve des documents relativement volumineux et que l'on peine à y trouver l'information réellement recherchée. Une fonction d'aide pourrait permettre de rapidement se situer dans le document à l'instar de Google qui propose une fonctionnalité similaire pour les documents cachés.

6.5.4 *Migration*

Les technologies associées à Internet sont en constante évolution et changent rapidement. Il est parfois nécessaire de migrer rapidement une application existante vers une nouvelle technologie. Les coûts de pareilles opérations sont souvent relativement élevés, surtout pour des serveurs importants.

Ces migrations sont aussi délicates, parce qu'il faut "basculer" l'ensemble du service sur la nouvelle technologie avec des mises en production brutales et difficiles à gérer. Les différentes technologies ne pouvant que difficilement cohabiter.

L'utilisation d'un proxy comme intermédiaire faisant le lien entre deux versions de l'application permet une migration par parties, et donc une mise en production plus en douceur.

6.6 *Cache +*

Nous avons présenté dans notre travail plusieurs paradigmes touchant à la gestion du cache. D'abord l'introduction d'une topologie basée sur les URL et les liens entre les documents au chapitre 3, ensuite au chapitre 6 le concept de cache métier et le calcul de la distance isotopique pour un mécanisme de bascule automatique dans le domaine courant. Enfin nous avons présenté ci-dessus une modélisation qui répond aux différents besoins que nous avons évoqués auparavant (modélisation du webgraph, persistance des index pour éviter des distorsions, annotations, etc.). Ces différents éléments nous permettent donc de mettre en place une gestion du cache avancée qu'il serait parfaitement possible de mettre en place sur notre plateforme.

Le protocole HTTP 1.1 spécifie des mécanismes à propos de la mise à jour des documents avec des directives donnant, en plus de la date des dernières modifications ou de création d'un document, des informations optionnelles pouvant préciser la validité (intervalle de temps) ou la date d'expiration du document, etc. Cependant, ces directives ne sont pas aisées à mettre en place du côté du serveur et sont de fait rarement utilisées, ou alors de manière souvent cavalière et rédhibitoire pour limiter abusivement la possibilité de caching des proxys.

De même que l'utilisateur peut compléter l'information sémantique d'un document en l'annotant, il peut, le cas échéant, proposer des informations de gestion pour le proxy. Ainsi pourra-t-il, sur un document de news quotidiennes, demander à ce que celui-ci soit renouvelé chaque jour dans le cache.

7 **Conclusion**

L'architecture I3 permet tout d'abord de mettre en place tous les concepts que nous avons élaborés au cours de notre travail : les différentes extensions en dimensions et l'interaction avec l'utilisateur au travers des sessions. A partir de la mise en place de ces différents concepts, nous avons montré qu'il était possible d'améliorer la fonction primaire d'un proxy, le cache, en proposant une gestion plus flexible, adaptable et prenant en compte plus de paramètres pertinents.

Mais l'architecture I3 permet aussi l'ouverture à une infinité d'applications. Le concept d'intermédiaire ou de proxlet ouvre autant de possibilités que l'ont fait les applets ou les servlets pour les ressources WWW. Nous avons suggéré quelques exemples dans ce qui précède, mais ne doutons pas que les possibilités sont infinies.

Notre architecture présente encore l'avantage d'être ouverte et il est donc aisé d'introduire de nouvelles facilités par la mise en place de nouveaux modules en s'appuyant sur le paradigme des proxlets.

SYNTHESE ET CONCLUSION

Depuis le début des années 90, l'Internet a énormément évolué aussi bien en nombre qu'en diversité de services disponibles. Nous distinguons trois étapes importantes dans l'évolution des services réseau, et plus particulièrement du Web, à l'instar des systèmes informatiques. Une première ère concentrée sur des super serveurs (mainframes), une deuxième période, avec une ascension forte des clients (client-serveur) et enfin une troisième phase concentrée sur le middleware – les proxys apparaissent comme la troisième voie de développement majeure pour les évolutions qui vont suivre l'introduction de serveurs toujours plus évolués et dynamiques et des clients incluant de plus en plus de fonctionnalités. A partir de ce constat, nous proposons de faire évoluer les proxys de simples intermédiaires passifs à une plateforme ouverte où pourront s'installer des services partagés par une ou plusieurs communautés d'utilisateurs.

En guise de conclusion, nous résumons ici le contexte et les enjeux de notre travail, puis nous retraçons brièvement la démarche qui a conduit notre recherche et enfin nous rappelons les différentes contributions que nous avons mises en place.

Au cours de notre travail, nous nous sommes autorisé des simplifications où nous avons ignoré certains problèmes. Nous résumons tout cela dans une brève critique.

Les proxys Internet offrent des perspectives quasi illimitées, c'est en tout cas ce que nous croyons. Pour le souligner, nous rapportons ici quelques prolongements possibles à notre travail et nous terminons par un bilan récapitulatif de l'ensemble de notre recherche.

1 Résumé du contexte

L'efficacité dans la mise à disposition d'informations pour l'utilisateur final reste un point fondamental de développement dans les applications réseau. L'étendue d'Internet couvre des millions de serveurs, de sous-réseaux et toutes les infrastructures annexes (routeur, firewall, LAN, WAN, liens trans-océaniques ou satellites, etc.). Alors que l'évolution d'Internet suit une courbe exponentielle [93] en termes de serveurs et d'utilisateurs, les infrastructures intermédiaires peinent à suivre cette croissance. La demande pour une augmentation des performances existe et elle se traduit chez les utilisateurs par la prise d'abonnements toujours plus avancés chez les Internet Providers (Câble, ADSL, etc.). Mais cette évolution est parcellaire : les différents éléments qui entrent en jeu dans la transmission des informations évoluent de manière disparate. Dans une chaîne de transport de l'information sur les réseaux, c'est le maillon le plus lent qui impose la vitesse de transmission.

L'ergonomie des services distribués sur les réseaux repose pour une bonne partie sur l'interactivité avec l'utilisateur final, qui elle-même dépend de la qualité des transmissions de l'information.

Les proxys permettent de réaliser une accélération de la vitesse apparente des transmissions. Ils apportent une solution qui améliore l'accès aux services sans avoir à faire évoluer l'ensemble de la chaîne de transport. L'insertion d'un proxy permet donc d'économiser des investissements élevés à plusieurs niveaux (par exemple le

dédoublage d'un lien trans-océanique représente des coûts colossaux). Cette technologie représente donc un enjeu majeur dans l'évolution d'Internet.

2 Résumé de la démarche

Pour que les proxys conservent leur importance stratégique qui se base sur la possibilité de pouvoir s'insérer avec un minimum d'impact dans les infrastructures existantes, nous nous sommes imposé dans ce travail de préserver l'usage des paradigmes existants. Dans le contexte du Web que nous avons choisi, il s'agit donc de préserver au maximum l'usage du protocole HTTP selon la norme du W3C.

Nous sommes partis de deux qualités du proxy : sa position privilégiée (plésiocentrique: au centre de la communauté) comme acteur dans les échanges entre une communauté et Internet, ainsi que sa perceptivité aux informations échangées et aux comportements des utilisateurs. Nous avons d'une part étendu la perceptivité, d'autre part nous avons exploité ces deux qualités pour développer trois points : la gestion du cache, le prefetching et la mise en place de services.

3 Contributions

Pour supporter les points que nous souhaitons développer, nous avons introduit deux paradigmes : le maintien de session et l'engnose.

Nous avons mis en place un mécanisme de maintien de session entre le proxy et l'agent client au travers des proxy-cookies. Ce paradigme basé sur les cookies standard nous permet d'abord d'identifier les utilisateurs et dans un deuxième temps de définir des profils. Sur cette base, nous pouvons développer des interactions entre l'utilisateur et le proxy ainsi qu'une perception étendue à de nouvelles informations statistiques.

Dans le cadre du Web sémantique, la deuxième génération proposée pour le Web, nous définissons l'engnose comme l'extension de la perceptivité du proxy aux informations sémantiques et aux ontologies.

Ces différents points nous permettent de développer les proxys dans quatre domaines : la gestion du cache, le prefetching, la mise en place de services intermédiaires et enfin la définition des proxlets qui supportent la mise en place de ces services.

3.1 *Gestion du cache*

Pour la gestion du cache, nous proposons d'abord la prise en compte de nouvelles dimensions par la constitution d'un espace topologique sémantique. Nous construisons cet espace en extrayant une sémantique de localisation des URL et des liens entre les documents. Pour cela, nous proposons la construction du Webgraph qui modélise la position des documents et les statistiques de parcours des clients. Cette nouvelle politique de gestion permet la prise en compte des relations sémantiques entre les ressources du Web, avec une sémantique induite par le comportement des utilisateurs et la position des documents.

Dans un deuxième temps, nous proposons une gestion du cache basée sur les informations qu'apporte l'engnose dans le contexte du Web sémantique. Cette nouvelle gestion nous permet, par un système de bascule automatique dans le domaine courant en fonction des ressources parcourues par l'utilisateur, de définir des caches métiers basés sur un partitionnement du cache. Enfin, la gestion du cache peut également se faire sur la valeur sémantique des ressources évaluée sur la base des annotations associées.

3.2 *Prefetching*

Notre modélisation des ressources dans le proxy nous permet d'élaborer un nouvel algorithme d'anticipation basé sur les chaînes de Markov. Alors que les propositions existantes peinent à prédire de longs chemins de parcours et à distinguer les ressources incluses, nous résolvons ce problème avec l'utilisation du Webgraph et la disponibilité de statistiques plus étendues.

La connaissance du domaine courant nous permet également de prédire plus fidèlement les prochaines ressources probables.

3.3 *Services intermédiaires*

La mise en place de services au niveau du proxy nous amène un certain nombre d'avantages. D'abord la situation plésiocentrique du proxy nous permet d'accumuler des connaissances partagées par une communauté et donc adaptées localement. Ensuite, l'exploitation de ces connaissances permet la mise en place de services répondant au mieux aux besoins des utilisateurs avec transparence. Nous avons ainsi exposé quelques exemples de services (aide à la navigation, intégration, filtrage, etc.) qui apportent de la valeur ajoutée aux proxys et aux informations qu'il contiennent.

3.4 *Proxlet*

Le paradigme de proxlet permet la généralisation du concept d'agent intermédiaire. En se basant sur ce paradigme, il est possible d'implémenter toute forme d'application au cœur de la communauté locale qui partage un proxy.

4 Critiques

La loi de Zipf permet de faire un certain nombre d'assertions sur les résultats probables qu'apportent les propositions que nous avons faites. Elle ne constitue cependant qu'une hypothèse qui a été validée pour de très larges échantillons, mais dont nous ne connaissons pas très bien la validité pour des populations réduites.

De plus, nous avons introduit des calculs relativement compliqués. Ainsi, dans le cadre du prefetching, si nous prenons l'exemple d'une prédiction de chemin de longueur 5 impliquant 30 documents, il nous faudra élever à la puissance 5 une matrice de 30x30, soit la somme de 108'000 produits⁴³. C'est parfaitement raisonnable, mais dans le cadre de centaines voir de milliers de requêtes à traiter cela peut constituer un étranglement.

Il en va de même pour toutes les évaluations sémantiques avec des tests de correspondance sur des vocabulaires qui peuvent être étendus.

Nous avons testé dans une implémentation de la plateforme I3 le fonctionnement des concepts de proxlet et de session avec succès. Nous n'avons pas eu l'occasion de valider, dans le cadre d'une utilisation à grande échelle (grand nombre d'utilisateurs), le fonctionnement de nos propositions. Même si ce que nous présentons reste donc relativement théorique, la mise en parallèle avec des techniques similaires (notamment pour la gestion du cache) nous permet d'affirmer pouvoir obtenir des résultats positifs sans pour autant faire état de mesures précises.

Nous avons évoqué à quelques endroits la possibilité pour les proxys de collaborer entre eux. Il existe même un protocole pour supporter cette collaboration: ICP [31]. Nous avons complètement omis cet aspect dans notre travail. Il est cependant certain que dans

⁴³ L'élevation au carré d'une matrice 30x30 entraîne 30x30x30=27'000 produits. L'élevation à la puissance 5 donne donc quatre fois ce nombre.

le but d'assurer un service de mobilité, les sessions seules ne suffiraient pas et qu'il faudrait également étudier l'opportunité d'utiliser les paradigmes existants ou le cas échéant de les étendre pour supporter les mécanismes que nous avons présentés.

5 Prolongements

C'est avec enthousiasme tout au long de notre travail que nous avons découvert le potentiel des proxys. Cependant, il a bien fallu à un moment donné cadrer le projet et mettre de côté certaines opportunités. De même, pour valider nos concepts il aurait été intéressant d'aller plus loin dans l'implémentation pour disposer d'une évaluation sur le terrain de nos propositions.

5.1 *Elargissement à d'autres rôles*

Le rôle des proxys est, dans le cas le plus général, de servir d'intermédiaire entre les utilisateurs locaux et "l'extérieur" et c'est le point de vue que nous avons adopté tout au long de notre travail. Ce rôle correspond à la fonction première pour laquelle les proxys ont été conçus. Cependant, sans en modifier le fonctionnement, les proxys peuvent être utilisés à d'autres fins. Un premier détournement qui a connu un certain succès est l'utilisation du proxy comme accélérateur de serveur HTTP. Mais on peut imaginer d'autres utilisations pour les proxys à partir de la plateforme que nous avons décrite.

- **Pour l'intégration.** Dans le cadre de structures non homogènes, le proxy peut jouer le rôle d'intermédiaire pour fédérer des applications distinctes. Bien entendu, le proxy n'est pas une plateforme concurrente pour des applications comme BEA WebLogic, mais dans le cadre d'intégrations simples, il pourrait se révéler un outil intéressant et facile à mettre en œuvre. Il serait par ailleurs intéressant d'étudier dans quelle mesure et comment les proxys pourraient être des intégrateurs.
- **Pour l'enrichissement.** Pour l'apport de nouvelles ressources dans des applications existantes, par exemple l'adjonction de documents multimédias, l'utilisation d'intermédiaires permet de facilement et simplement insérer ces nouvelles données sans avoir à modifier l'existant. Par exemple, dans un service mettant à disposition les fiches documentaires pour une phonothèque, un intermédiaire pourrait facilement "intercepter" les identifiants des fiches et insérer une référence sur la ressource sonore elle-même.
- **Pour les migrations.** Malgré sa jeunesse, le Web a subi déjà de nombreuses mutations technologiques. Ainsi des services se doivent d'évoluer pour différentes raisons⁴⁴ ce qui peut se révéler périlleux. L'utilisation d'intermédiaires pourrait permettre une intégration des changements par palier.
- **Pour le Peer-2-Peer** (architecture dans laquelle tous les clients jouent systématiquement le rôle de serveur). Les applications de P2P reposent soit sur un index centralisé (ex. Napster), soit sur une structure complètement décentralisée avec des index disséminés chez les clients (ex: BearShare). Dans le cadre d'index disséminés, la recherche de ressources est une opération rendue compliquée par la granularité (n clients donc n index) quel que soit le modèle proposé (voir [94] pour plus de détails sur les techniques d'indexation). De plus, les transactions entre des clients qui se trouvent derrière des firewalls requièrent le support d'un proxy pour le transport des informations. Dans ce type d'applications, l'utilisation d'intermédiaires capables de centraliser partiellement ces index impliquant ainsi la granularité augmenterait les performances.

⁴⁴ Voir à ce propos le cas de Yahoo: <http://public.yahoo.com/~radwin/talks/yahoo-phpcon2002.htm>

Ce ne sont que quelques propositions d'ouverture à de nouveaux champs d'application qui nous sont apparues au cours de notre travail. Le concept d'intermédiaire est ouvert et mérite encore d'être exploré.

5.2 *Élargissement à d'autres protocoles*

Dans notre travail, nous n'avons abordé que les cas d'utilisation de proxys avec le protocole HTTP. Ce protocole est très largement utilisé, dans un nombre croissant d'applications. Cependant il existe d'autres cas de figure pour la transmission sur le réseau de données. Par exemple le protocole ftp, pour lequel d'ailleurs les premières applications de caching proxy ont été développées. Nous allons ici aborder quelques possibilités pour la mise en application de caching proxys sur la base d'autres protocoles que HTTP.

Mais rappelons d'abord les principales caractéristiques du protocole http : les objets sont clairement délimités, les transactions se limitent aux opérations de la forme question-réponse. Dans ce cadre, le cache peut donc facilement associer une réponse avec une requête.

5.2.1 *Streaming*

Le streaming est en général utilisé pour les applications vidéo et audio. Dans ces deux domaines, les objets impliqués sont de taille importante et du point de vue temporel proposent presque exclusivement une lecture séquentielle. Ceci implique la possibilité de consulter le document avant qu'il ne soit complètement disponible du côté de l'agent, ce qui se traduit par une lecture au fur et à mesure que la séquence se complète avec de nouvelles données téléchargées.

Plutôt que de définir le streaming comme le fait d'amener la radio ou la TV sur le Web, nous dirons que le streaming est l'envoi à un agent de données fortement temporelles et ordonnées de manières séquentielles.

Cette définition reste cependant floue. Elle couvre bien toutes les applications de streaming que nous connaissons, mais conceptuellement, le fait de télécharger un objet audio ou vidéo par le protocole HTTP et de permettre à l'agent de le "jouer" au fur et à mesure, rentre également dans le champ défini ici. De plus, dans ce dernier exemple, nous nous retrouvons complètement dans une transaction standard et les mécanismes de caching "classiques" conviennent parfaitement.

Il existe cependant deux problèmes qui apparaissent presque systématiquement dans le streaming et qui échappent au caching "classique": d'une part la négociation du débit et de la qualité, d'autre part l'acquisition par segments.

5.2.2 *Telnet*

Malgré l'enthousiasme soulevé par les technologies du Web et la quasi généralisation de l'usage de celles-ci, il existe encore des applications qui ne sont pas basées sur le protocole HTTP. Dans notre travail, nous nous sommes intéressés à un cas particulier, l'utilisation de Telnet dans des applications multi-utilisateurs, en l'occurrence une catégorie de jeux: les MUD (Multi User Dungeon). Dans cette application, les objets sont transportés sur une connexion de manière indistincte. De plus, les objets ne sont pas envoyés à la requête du client mais en fonction de ses actions ou d'activités apparaissant dans le jeu. Il est par conséquent difficile d'effectuer du caching. Pourtant la répétition de l'information est importante et l'utilisation d'une telle technique permettrait des économies substantielles. Pour permettre l'introduction d'un caching proxy, nous avons alors proposé d'introduire des directives Telnet (options telnet TOPT). Ainsi, nous pouvons découper le flux pour "marquer" les objets ou envoyer des identifiants et ne pas transférer l'objet.

L'utilisation d'un intermédiaire nous permet également, comme suggéré plus haut, d'enrichir ce jeu au départ purement textuel, en y ajoutant des images, des animations, etc.

Ce dernier exemple peut paraître marginal, mais ce type d'applications draine aujourd'hui déjà des millions d'utilisateurs et les performances jouent un rôle majeur dans la "jouabilité".

6 Bilan

La plateforme I3 avec la possibilité de mettre en place très rapidement des services complémentaires à la navigation sur Internet nous semble très prometteuse. Nous pensons que le paradigme des intermédiaires suscitera encore de nombreux développements, que ce soit par la nécessité de gagner en performances ou pour profiter de l'opportunité du plésiocentrisme pour apporter de nouveaux services.

Les premiers développements d'Internet se sont surtout concentrés sur les techniques de mise à disposition d'informations. On a rapidement constaté, avec la prolifération incontrôlée qui a suivi, que cela ne suffisait pas. Le Web sémantique propose alors de travailler sur la mise en valeur de ces informations. Un troisième temps se concentrera sur l'exploitation de cette information. Dans ce contexte, nous assisterons à des développements au niveau des clients, mais aussi certainement des intermédiaires, et l'architecture ainsi que les concepts que nous proposons constituent une excellente base à de tels développements.

Enfin, l'enrichissement des services (jeux, librairies en lignes, etc.) ne pourra pas se faire sans un accroissement des performances. Cet accroissement ne pourra pas se faire uniquement avec une augmentation de toutes les infrastructures réseau. Le déplacement des ressources dans le voisinage des utilisateurs semble incontournable. Les paradigmes que nous avons exposés, par leur flexibilité et leur généricité, permettent d'intégrer facilement de nouvelles applications. On pourra donc facilement exploiter ce travail dans de nouveaux contextes comme nous l'avons brièvement présenté ci-dessus.

BIBLIOGRAPHIE

- [1] E. P. Markatos, "A Cash-based Approach to Caching Web Documents," ICS-FORTH, 1998.
- [2] R. P. Wooster and M. Abrams, "Proxy Caching That Estimates Page Load Delays," presented at the 6th International World Wide Web Conference, 1997.
- [3] L. Cherkasova, "Improving WWW Proxies Performance with Greedy-Dual-Size-Frequency," HP ed, 1998.
- [4] C. Aggarwal, J. L. Wolf, and P. S. Yu, "Caching on the World Wide Web," presented at IEEE Transactions on knowledge and data engineering, 1999.
- [5] M. Reddy and G. P. Fletcher, "Intelligent web caching using document life histories: A comparison with existing cache management techniques," presented at the Third International WWW Caching Workshop, Manchester, 1998.
- [6] T. Berners-Lee, "Information Management: A Proposal," CERN, Internal report for CERN March 1989, May 1990.
- [7] M. Levene and G. Loizou, "Computing the entropy of user navigation in the web," Department of Computer Science, University College London, 22 Feb 2000.
- [8] M. F. Arlitt, R. Friedrich, and T. Jin, "Performance Evaluation of Web Proxy Cache Replacement Policies," <http://citeseer.nj.nec.com/arlitt98performance.html> ed: HP, 1999.
- [9] E. Cohen, B. Krishnamurthy, and J. Rexford, "Improving End-to-End Performance of the Web Using Server Volumes and Proxy Filters," in *SIGCOMM*, 1998, pp. 241-253.
- [10] E. Cohen, B. Krishnamurthy, and J. Rexford, "Efficient Algorithm for Predicting Requests to Web Servers," presented at IEEE INFOCOM, 1999.
- [11] N. Ashish, C. F. Knoblock, and C. Shahabi, "Intelligent Caching for Information Mediators: A KR Based Approach," presented at the 5th Knowledge Representation Meets Databases Workshop (KRDB), Seattle, 1998.
- [12] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "RFC2616: Hypertext Transfer Protocol -- HTTP/1.1," <ftp://ftp.isi.edu/in-notes/rfc2616.txt> ed: RFC-editor, 1999.
- [13] T. Berners-Lee, "Document caching," Archives: <http://www.w3.org/DesignIssues/Caching.html> ed: CERN-W3C, 1990.
- [14] "NCSA Mosaic Home Page," <http://archive.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html> ed: NCSA.
- [15] P. B. Danzig, R. S. Hall, and M. F. Schwartz, "A Case for Caching File Objects Inside Internetworks," presented at SIGCOMM, 1993.
- [16] A. Luotonen and K. Altis, "World-Wide Web proxies," *Computer Networks and ISDN Systems*, vol. 27, pp. 147-154, 1994.
- [17] C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, and M. F. Schwartz, "The Harvest information discovery and access system," *Computer Networks and ISDN Systems*, vol. 28, pp. 119-125, 1995.
- [18] "Akamai Content Delivery," Web documentation: http://www.akamai.com/en/html/services/content_delivery.html ed: Akamai, 2002.

- [19] R. Barret, P. P. Maglio, and D. C. Kellem, "How to Personalize the Web," presented at ACM Conference on Human Factors in Computing Systems (CHI '97), Atlanta, GA, 1997.
- [20] R. Barret and P. P. Maglio, "Intermediaries: new places for producing and manipulating web content," presented at the 7th International World Wide Web Conference (WWW7), Brisbane, Australia, 1998.
- [21] I. Cooper and J. Dilley, "RFC 3143: Known HTTP Proxy/Caching Problems," <ftp://ftp.isi.edu/in-notes/rfc3143.txt> ed: RFC-editor, 2001.
- [22] M. Abrams, C. R. Standridge, G. Abdulla, S. Williams, and E. A. Fox, "Caching Proxies: Limitations and Potentials," presented at the 4th International Conference on the World-Wide Web, Boston, 1995.
- [23] R. Orfali, D. Harkey, and J. Edwards, *Client/Server Survival Guide*, Third ed: Wiley, 1999.
- [24] E. Buff, Y. Dennebouy, P. Rochat, and S. Spaccapietra, "Mass Archiving of Multimedia Data," presented at Advances in Multimedia and Databases for the New Century: A Swiss/Japanese Perspective, 1999.
- [25] V. Holmedahl, B. Smith, and T. Yang, "Cooperative Caching of Dynamic Content on a Distributed Web Server," presented at the 7th IEEE International Symposium on High Performance Distributed Computing, 1998.
- [26] M. Rabinovich, "Issues in Web Content Replication," *IEEE Data Engineering Bulletin*, vol. 21, pp. 21-29, 1998.
- [27] M. R. Boyns, "Muffin," <http://muffin.doit.org/> ed: San Diedo State University.
- [28] H. Chen, M. Abrams, T. Johnson, A. Mathur, I. Anwar, and J. Stevenson, "Wormhole Caching with HTTP PUSH Method for a Satellite-Based Web Content Multicast and Replication System," presented at the 4th International Web Caching Workshop, San Diego, 1999.
- [29] D. Adams, *Programming Jabber*. O'Reilly, 2001.
- [30] B. Williams, "Transparent Web Caching Solutions," presented at the 3rd International WWW Caching Workshop, Manchester, England, 1998.
- [31] D. Wessels and K. Claffy, "RFC2186: Internet Cache Protocol (ICP), version 2," <ftp://ftp.rfc-editor.org/in-notes/rfc2186.txt> ed: RFC-editor, 1997.
- [32] J. Touch, "The LSAM proxy cache - a multicast distributed virtual cache," presented at the 3rd Int. WWW Caching Workshop, San Diego, 1998.
- [33] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell, "A Hierarchical Internet Object Cache," presented at the USENIX Technical Conference, 1996.
- [34] M. Makpangou and E. Béranguier, "Relais: Un protocole de maintien de cohérence de caches Web coopérants," presented at NoTeRe '97, Pau, France, 1997.
- [35] P. Rodriguez, K. W. Ross, and E. Biersack, "Improving the WWW: Caching or Multicast?," *Computer Networks and ISDN Systems*, vol. 30, pp. 2223-2243, 1998.
- [36] E. Biersack, "Efficient Distribution of Web Documents over the Internet: Caching or Multicast?," Eurecom, 1999.
- [37] W. R. Stevens, *TCP/IP Illustrated, TCP For Transactions, HTTP, NNTP, and the UNIX Domain Protocol*, vol. 3, 1996 ed: Addison-Wesley, 1996.
- [38] "Original design issues," Archives: <http://www.w3.org/DesignIssues> ed: W3C.
- [39] T. Berners-Lee, "Link Types," Archives: <http://www.w3.org/DesignIssues/LinkTypes.html> ed: W3C, 1990.
- [40] T. Berners-Lee, "Multiuser considerations," Archives: <http://www.w3.org/DesignIssues/Multiuser.html> ed: W3C, 1990.

- [41] T. Berners-Lee, "Semantic Web Road map," <http://www.w3.org/DesignIssues/Semantic.html> ed: W3C, 1998.
- [42] "The Semantic Web Community Portal," <http://www.semanticweb.org> ed: SemanticWeb.org.
- [43] J. Chuang, S. Kafka, and K. Norlen, "Efficiency and Performance of Web Cache Reporting Strategies," presented at IEEE International Workshop on Data Semantics in Web Information Systems, Singapore, 2002.
- [44] J. Mogul and P. Leach, "RFC 2227: Simple Hit-Metering and Usage-Limiting for HTTP," <ftp://ftp.isi.edu/in-notes/rfc2396.txt> ed: RFC-editor, 1997.
- [45] I. I. United States Internet Council, "State of the Internet 1999 and 2000," <http://www.usic.org/> ed, 2000.
- [46] P. Cao and S. Irani, "Cost-Aware WWW Proxy Caching Algorithms," presented at Usenix Symposium on Internet Technologies and Systems: USITS, Monterey, CA, 1997.
- [47] S. Irani, "Page Replacement with Multi-Size Pages and Applications to Web Caching," presented at the 29th Annual ACM Symposium on Theory of Computing, 1997.
- [48] J. Shim, P. Scheuermann, and R. Vingralek, "Proxy Cache Design: Algorithms, Implementation and Performance," *IEEE Transactions on Knowledge and Data Engineering*, 1999.
- [49] P. Lorenzetti, L. Rizzo, and L. Vicisano, "Replacement policies for a proxy cache," Università di Pisa, Dec 1996.
- [50] J. Shim, P. Scheuermann, and R. Vingralek, "A Unified Algorithm for Cache Replacement and Consistency in Web Proxy Servers," presented at International Workshop on the Web and Databases (WebDB98), 1998.
- [51] M. F. Arlitt, L. Cherkasova, J. Dille, R. Friedrich, and T. Jin, "Evaluating Content Management Techniques for Web Proxy Caches," <http://www.hpl.hp.com/techreports/98/HPL-98-173.html> ed: HP, 1999.
- [52] E. Cohen and H. Kaplan, "Caching Documents with Variable Sizes and Fetching Costs: An LP-Based Approach," *Algorithmica*, vol. 32, pp. 459-466, 1999.
- [53] S. Williams, M. Abrams, C. R. Standridge, G. Abdulla, and E. A. Fox, "Removal Policies in Network Cache for World-Wide-Web Documents," presented at ACM SIGCOMM, 1996.
- [54] C. Aggarwal and P. S. Yu, "On Disk Caching of Web Objects in Proxy Servers," presented at CIKM, 1997.
- [55] X. Chen and X. Zhang, "Coordinated Data Prefetching by Utilizing Reference Information at Both Proxy and Web Servers," presented at PAWS, 2001.
- [56] D. Duchamp, "Prefetching hyperlinks," presented at the 2nd USENIX Symposium on Internet Technologies and Systems (USITS '99), Boulder, CO, 1999.
- [57] K. M. Curewitz, P. Krishnan, and J. S. Vitter, "Practical prefetching via data compression," presented at ACM-SIGMOD Conference on Management of Data, Washington DC, 1993.
- [58] A. Kraiss and G. Weikum, "Integrated document caching and prefetching in storage hierarchies based on Markov-chain predictions," *VLDB Journal: Very Large Data Bases*, vol. 7, pp. 141-162, 1998.
- [59] L. Fan, P. Cao, W. Lin, and Q. Jacobson, "Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance," in *Measurement and Modeling of Computer Systems*, 1999, pp. 178-187.

- [60] A. Venkataramani, P. Yalagandula, R. Kokku, S. Sharif, and M. Dahlin, "The potential costs and benefits of long term prefetching for content distribution," University of Texas at Austin TR-01-13, 2001.
- [61] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications," presented at IEEE Infocom, New York, 1999.
- [62] C. R. Cunha, A. Bestavros, and M. E. Crovella, "Characteristics of WWW Client-based Traces," Boston University Technical report BU-CS-95-010, Jul 1995.
- [63] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajogopalan, and A. S. Tomkins, "The Web as a graph: measurements, models and methods," presented at the 5th Annual International Computing and Combinatorics Conference, 1999.
- [64] T. Berners-Lee, R. Fielding, and L. Masinter, "RFC2396: Uniform Resource Identifiers (URI): Generic Syntax," <ftp://ftp.isi.edu/in-notes/rfc2396.txt> ed: RFC-editor, 1998.
- [65] H. Liang, "A URL-String-Based Algorithm for Finding WWWMirror Hosts," <http://citeseer.nj.nec.com/463200.html> ed, 2001.
- [66] H. Ritter, T. Martinetz, and K. Schulten, *Neural Computation and Self-Organizing Maps - An Introduction*. New York: Addison-Wesley, 1992.
- [67] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, pp. 59-69, 1982.
- [68] R. Miikkulainen, "Self-Organizing Process Based on Lateral Inhibition and Synaptic Resource Redistribution," in *Artificial Neural Networks*, vol. I, T. Kohonen, K. Makisara, O. Simula, and J. Kangas, Eds.: North-Holland, 1991, pp. 415-420.
- [69] B. D. Malamud and D. L. Turcotte, "Wavelet analysis of mars mola topography," *Lunar and Planetary Science*, Mar 14 2000.
- [70] M. Kurcewicz, W. Sylwestrzak, and A. Wierzbicki, "A Filtering Algorithm for Proxy Caches," presented at the 3rd International WWW Caching Workshop, Manchester, England, 1998.
- [71] A. Ruegg, *Processus Stochastiques*, vol. 6. Lausanne: Presses Polytechniques Romandes, 1989.
- [72] D. Kristol and L. Montulli, "RFC 2965: HTTP State Management Mechanism," <ftp://ftp.rfc-editor.org/in-notes/rfc2965.txt> ed: RFC-Editor, 2000.
- [73] S. Handschuh, S. Staab, and A. Mädche, "CREAM - Creating relational metadata with a component-based, ontology driven annotation framework," presented at K-CAP, 2001.
- [74] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, May 17 2001.
- [75] S. DeRose, E. Maler, and D. Orchard, "XML Linking Language (XLink) Version 1.0," <http://www.w3.org/TR/xlink/> ed, 2001.
- [76] J. Kahan, M.-R. Koivunen, E. Prud'Hommeaux, and R. R. Swick, "Annotea: An open RDF infrastructure for shared web annotations," presented at the 10th International World Wide Web Conference, Hong Kong, 2001.
- [77] M.-R. Koivunen, D. Brickley, J. Kahan, E. Prud'Hommeaux, and R. Swick, "The W3C collaborative web annotation project ... or how to have fun while building an RDF infrastructure," <http://www.w3.org/2002/02/collaboration/annotation/papers/annotationinfrastructure> ed, 2000.
- [78] ESI, "ESI," <http://www.esi.org/> ed.

- [79] Oracle, "ESI," http://otn.oracle.com/sample_code/products/ias/web_cache/htdocs/jesi/jesidemo.html ed.
- [80] J. Marsh and D. Orchard, "XML Inclusions (XInclude) Version 1.0," <http://www.w3.org/TR/xinclude/> ed: W3C, 2002.
- [81] P. Grosso, E. Maler, J. Marsh, and N. Walsh, "XPointer Framework," <http://www.w3.org/TR/xptr-framework/> ed: W3C, 2002.
- [82] M. Reisslein, F. Hartanto, and K. W. Ross, "Interactive video streaming with proxy servers," *Information Sciences*, vol. 140, pp. 3-31, 2002.
- [83] J. Hendler, "Foreword - Knowledge is Power Again !," in *Towards The Semantic Web, Ontology-Driven Knowledge Management*: Wiley, 2003, pp. xii.
- [84] J. Davies, D. Fensel, and F. V. Harmelen, *Towards The Semantic Web, Ontology-Driven Knowledge Management*: Wiley, 2003.
- [85] E. T. Hall, *La dimension cachée (The Hidden Dimension)*: Seuil, 1971, 1966.
- [86] S. Mastrogioacomo, "Utilisation de zones de travail partagées asynchrones pour améliorer la compréhension mutuelle dans les groupes de projets distribués.," in *H.E.C. Lausanne: Université de Lausanne*, 2002.
- [87] C. Fuchs, "Linguistique (notions de base)," in *Encyclopédie Universalis*, Universalis, Ed., 2001.
- [88] S. Spaccapietra and C. Parent, "Erc+: An Object Based Entity-Relationship Approach," in *Conceptual Modelling, Databases and CASE: an Integrated View of Information Systems Development*, 1992.
- [89] V. F. Almeida, M. G. Cesario, R. C. Fonseca, W. J. Meira, and C. D. Murta, "Analyzing the Behavior of a Proxy Server in Light of Regional and Cultural Issues," presented at the 3rd International WWW Caching Workshop, Manchester, England, 1998.
- [90] J. Dubois, *Dictionnaire de linguistique et des sciences du langage*. Paris: Larousse, 1994.
- [91] M. Sarkar and M. H. Brown, "Graphical Fisheye Views," *Communications of the ACM*, vol. 37, pp. 73-84, 1994.
- [92] H. Weinreich, H. Obendorf, and W. Lamersdorf, "The Look of the Link - Concepts for the User Interface of Extended Hyperlinks," presented at ACM Conference on Hypertext and Hypermedia (Hypertext '01), Aarhus, Denmark, 2001.
- [93] ISC, "ISC, Internet Software Consortium," <http://www.isc.org/> ed.
- [94] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Puceva, R. Schmidt, and J. Wu, "Advanced Peer-to-Peer Networking: The P-Grid System and its Applications," *PIK Journal - Praxis der Informationsverarbeitung und Kommunikation, Special Issue on P2P Systems*, 2002.