HTTP

Le guide définitif

www.it-ebooks.info

HTTP

Le guide définitif David Gourley et Brian Totty

avec Marjorie Sayer, Sailu Reddy et Anshu Aggarwal

Pékin • Cambridge • Farnham • Cologne • Paris • Sébastopol • Taipei • Tokyo

HTTP : Le guide définitif par David Gourley et Brian Totty avec Marjorie Sayer, Sailu Reddy et Anshu Aggarwal

Publié par O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Les livres d'O'Reilly Media, Inc. peuvent être achetés à des fins éducatives, commerciales ou promotionnelles. Des éditions en ligne sont également disponibles pour la plupart des titres (safari.oreilly.com). Pour plus d'informations, contactez notre service commercial aux entreprises et aux institutions : (800) 998-9938 ou corporate@oreilly.com .

Rédactrice en chef : Linda Mui

Monteuse de production : Rachel Wheeler

Concepteur de la couverture : Ellie Volckhausen

Architectes d'intérieur :

Histoire de l'imprimerie : David Futato et Melanie Wang

Septembre 2002 : Première édition.

Ce livre utilise RepKover™, une reliure à plat durable et flexible.

ISBN-10 : 1-56592-509-2

ISBN-13 : 978-1-56592-509-0

[C] [01/08]

Table des matières

Préface

Le protocole de transfert hypertexte (HTTP) est le protocole utilisé par les programmes pour communiquer sur le Web. Ses applications sont nombreuses, mais il est surtout connu pour ses échanges bidirectionnels entre navigateurs et serveurs web.

HTTP a commencé comme un simple protocole, et vous pourriez penser qu'il n'y a pas grand-chose à dire à son sujet. Et pourtant, vous voilà, un livre de 900 g entre les mains. Si vous vous demandez comment nous avons pu écrire 650 pages sur HTTP, jetez un œil à la table des matières. Ce livre n'est pas seulement un manuel de

référence sur les en-têtes HTTP ; c'est une véritable bible de l'architecture web.

Dans ce livre, nous tentons de décrypter les règles interdépendantes et souvent mal comprises du protocole HTTP, et vous proposons une série de chapitres thématiques qui expliquent tous les aspects du protocole. Tout au long du livre, nous prenons soin d'expliquer le « pourquoi » du protocole HTTP, et pas seulement le « comment ». Et pour vous éviter de perdre du temps à chercher des références, nous expliquons de nombreuses technologies non HTTP essentielles au fonctionnement des applications HTTP. Vous trouverez la référence alphabétique des en-têtes (qui constitue la base de la plupart des textes HTTP conventionnels) dans une annexe bien organisée. Nous espérons que cette conception vous facilitera l'utilisation du protocole HTTP.

Ce livre s'adresse à tous ceux qui souhaitent comprendre HTTP et l'architecture sous-jacente du Web. Les ingénieurs logiciels et matériels peuvent l'utiliser comme une référence cohérente pour HTTP et les technologies Web associées. Les architectes système et les administrateurs réseau peuvent l'utiliser pour mieux comprendre la conception, le déploiement et la gestion d'architectures Web complexes. Les ingénieurs et analystes de performance peuvent tirer profit des sections sur la mise en cache et l'optimisation des performances. Les professionnels du marketing et du conseil pourront utiliser l'orientation conceptuelle pour mieux comprendre le paysage des technologies Web.

Cet ouvrage illustre les idées reçues, propose des astuces, fournit des documents de référence pratiques et constitue une introduction facile aux spécifications de normes complexes et complexes. Dans un seul ouvrage, nous détaillons les technologies essentielles et interdépendantes qui font fonctionner le Web.

Ce livre est le fruit d'un travail colossal mené par de nombreuses personnes passionnées par les technologies Internet. Nous espérons qu'il vous sera utile.

Exemple d'exécution : la quincaillerie de Joe

Plusieurs de nos chapitres incluent un exemple concret d'une quincaillerie et d'un magasin de bricolage en ligne, « Joe's Hardware », afin de démontrer des concepts technologiques. Nous avons créé un véritable site web pour cette boutique (http://www.joes-hardware.com) afin que vous puissiez tester certains exemples du livre. Nous maintiendrons ce site web tant que ce livre sera imprimé.

Guide chapitre par chapitre

Ce livre contient 21 chapitres, divisés en 5 parties logiques (chacune avec un thème technologique), et 8 annexes utiles contenant des données de référence et des aperçus des technologies connexes :

Partie I, HTTP : les fondements du Web

Partie II, Architecture HTTP

Partie III, Identification, autorisation et sécurité

Partie IV, Entités, codages et internationalisation

Partie V, Publication et distribution de contenu

Partie VI, Annexes

La première partie, HTTP : le fondement du Web, décrit la technologie de base de HTTP, le fondement du Web, en quatre chapitres :

• Le chapitre 1, Présentation de HTTP, est un aperçu rapide de HTTP.

• Le chapitre 2, URL et ressources, détaille les formats des localisateurs de ressources uniformes (URL) et les différents types de ressources qu'ils désignent sur Internet. Il décrit également l'évolution vers les noms de ressources uniformes (URN).

• Chapitre 3, Messages HTTP, détaille comment les messages HTTP transportent le contenu Web.

• Le chapitre 4, Gestion des connexions, explique les règles et comportements souvent mal compris et mal documentés pour la gestion des connexions HTTP.

La deuxième partie, Architecture HTTP, met en avant le serveur HTTP, le proxy, le cache, la passerelle et les applications robotisées, qui constituent les éléments constitutifs de l'architecture des systèmes web. (Les navigateurs web constituent bien sûr un autre élément constitutif, mais ils ont déjà été traités en détail dans la première partie du livre.) La deuxième partie comprend les six chapitres suivants :

• Le chapitre 5, Serveurs Web, donne un aperçu des architectures de serveurs Web.

• Le chapitre 6, Proxies, explore les serveurs proxy HTTP, qui sont des serveurs intermédiaires qui agissent comme des plates-formes pour les services et les contrôles HTTP.

• Le chapitre 7, Mise en cache, se penche sur la science des caches Web, des dispositifs qui améliorent les performances et réduisent le trafic en créant des copies locales de documents populaires.

• Le chapitre 8, Points d'intégration : passerelles, tunnels et relais, explique les passerelles et les serveurs d'applications qui permettent à HTTP de fonctionner avec des logiciels qui parlent différents protocoles, y compris les protocoles cryptés Secure Sockets Layer (SSL).

• Le chapitre 9, Robots Web, décrit les différents types de clients qui envahissent le Web, y compris les navigateurs omniprésents, les robots et les araignées, ainsi que les moteurs de recherche.

• Le chapitre 10, HTTP-NG, parle des développements HTTP encore en cours : le protocole HTTP-NG.

La troisième partie, « Identification, autorisation et sécurité », présente un ensemble de techniques et de technologies permettant de suivre l'identité, de renforcer la sécurité et de contrôler l'accès au contenu. Elle comprend les quatre chapitres suivants :

• Le chapitre 11, Identification du client et cookies, parle des techniques permettant d'identifier les utilisateurs afin que le contenu puisse être personnalisé en fonction du public d'utilisateurs.

• Le chapitre 12, Authentification de base, met en évidence les mécanismes de base permettant de vérifier l'identité des utilisateurs. Il examine également l'interface entre l'authentification HTTP et les bases de données.

• Le chapitre 13, Authentification Digest, explique l'authentification Digest, une amélioration complexe proposée à HTTP qui offre une sécurité considérablement améliorée.

• Le chapitre 14, HTTP sécurisé, est un aperçu détaillé de la cryptographie Internet, des certificats numériques et de SSL.

La partie IV, Entités, Codages et Internationalisation, se concentre sur le corps des messages HTTP (qui contient le contenu web) et sur les normes web qui décrivent et manipulent le contenu stocké dans ces corps. Elle comprend trois chapitres :

• Le chapitre 15, Entités et codages, décrit la structure du contenu HTTP.

• Le chapitre 16, Internationalisation, examine les normes Web qui permettent aux utilisateurs du monde entier d'échanger du contenu dans différentes langues et jeux de caractères.

• Le chapitre 17, Négociation de contenu et transcodage, explique les mécanismes de négociation de contenu acceptable.

La partie V, Publication et diffusion de contenu, aborde les technologies de publication et de diffusion de contenu web. Elle comprend quatre chapitres :

• Le chapitre 18, Hébergement Web, décrit les façons dont les gens déploient des serveurs dans des environnements d'hébergement Web modernes et la prise en charge HTTP pour l'hébergement Web virtuel.

• Le chapitre 19, Systèmes de publication, traite des technologies permettant de créer du contenu Web et de l'installer sur des serveurs Web.

• Le chapitre 20, Redirection et équilibrage de charge, examine les outils et techniques de distribution du trafic Web entrant entre un ensemble de serveurs.

• Le chapitre 21, Journalisation et suivi de l'utilisation, couvre les formats de journaux et les questions courantes.

La partie VI, Annexes, contient des annexes de référence utiles et des didacticiels sur les technologies connexes :

• L'annexe A, Schémas d'URI, résume les protocoles pris en charge par les schémas d'identifiant de ressource uniforme (URI).

• L'annexe B, Codes d'état HTTP, répertorie de manière pratique les codes de réponse HTTP.

• L'annexe C, Référence d'en-tête HTTP, fournit une liste de référence des champs d'en-tête HTTP.

• L'annexe D, Types MIME, fournit une liste complète des types MIME et explique comment les types MIME sont enregistrés.

• L'annexe E, Codage en base 64, explique le codage en base 64, utilisé par l'authentification HTTP.

• L'annexe F, Digest Authentication, donne des détails sur la façon de mettre en œuvre divers schémas d'authentification dans HTTP.

• L'annexe G, Balises de langue, définit les valeurs des balises de langue pour les en-têtes de langue HTTP.

• L'annexe H, Registre des jeux de caractères MIME, fournit une liste détaillée des encodages de caractères utilisés pour la prise en charge de l'internationalisation HTTP.

Chaque chapitre contient de nombreux exemples et pointeurs vers des documents de référence supplémentaires.

Conventions typographiques

Dans ce livre, nous utilisons les conventions typographiques suivantes :

Italique

Utilisé pour les URL, les fonctions C, les noms de commandes, les types MIME, les nouveaux termes lorsqu'ils sont définis et l'emphase

Largeur constante

Utilisé pour la sortie informatique, le code et tout texte littéral

Largeur constante en gras

Utilisé pour la saisie utilisateur

Commentaires et questions

Veuillez adresser vos commentaires et questions concernant ce livre à l'éditeur :

O'Reilly & Associés, Inc.

1005, route de Gravenstein Nord

Sébastopol, CA 95472

(800) 998-9938 (aux États-Unis ou au Canada)

(707) 829-0515 (international/local)

(707) 829-0104 (fax)

Ce livre dispose d'une page web contenant des errata, des exemples et des informations complémentaires. Vous pouvez y accéder à l'adresse suivante :

http://www.oreilly.com/catalog/httptdg/

Pour commenter ou poser des questions techniques sur ce livre, envoyez un e-mail à :

bookquestions@oreilly.com

Pour plus d'informations sur les livres, les conférences, les centres de ressources et le réseau O'Reilly, consultez le site Web d'O'Reilly à l'adresse :

http://www.oreilly.com

Remerciements

Ce livre est le fruit du travail de plusieurs personnes. Les cinq auteurs tiennent à remercier quelques personnes pour leur contribution significative à ce projet.

Pour commencer, nous tenons à remercier Linda Mui, notre éditrice chez O'Reilly. Linda a rencontré David et Brian pour la première fois en 1996, et elle a peaufiné et orienté plusieurs concepts pour aboutir au livre que vous tenez aujourd'hui entre vos mains. Elle a également contribué à maintenir notre groupe d'auteurs débutants dans une direction cohérente et un calendrier de progression (voire rapide). Mais surtout, Linda nous a donné l'opportunité de créer ce livre. Nous lui en sommes très reconnaissants.

Nous tenons également à remercier plusieurs personnes exceptionnellement brillantes, compétentes et bienveillantes qui ont consacré une énergie remarquable à la relecture, aux commentaires et à la correction des versions préliminaires de ce livre. Parmi elles, citons Tony Bourke, Sean Burke, Mike Chowla, Shernaz Daver, Fred Douglis, Paula Ferguson, Vikas Jha, Yves Lafon, Peter Mattis, Chuck Neerdaels, Luis Tavera, Duane Wessels, Dave Wu et Marco Zagha. Leurs points de vue et suggestions ont considérablement amélioré ce livre.

Rob Romano, d'O'Reilly, a créé la plupart des magnifiques illustrations de ce livre. Ce dernier contient un nombre exceptionnel d'illustrations

détaillées qui éclairent des concepts subtils. Nombre de ces illustrations ont été créées avec le plus grand soin et maintes fois retouchées. Si une image vaut mille mots, Rob a enrichi ce livre de centaines de pages.

Brian tient à remercier personnellement tous les auteurs pour leur dévouement à ce projet. Ils ont consacré énormément de temps à la réalisation du premier traitement détaillé et accessible du protocole HTTP. Mariages, naissances, projets professionnels ambitieux, startups et écoles supérieures ont été au rendez-vous, mais les auteurs ont su s'unir pour mener à bien ce projet. Nous pensons que le résultat est à la hauteur du travail acharné de chacun et, surtout, qu'il offre un service précieux. Brian tient également à remercier les employés d'Inktomi pour leur enthousiasme, leur soutien et leurs connaissances approfondies sur l'utilisation du protocole HTTP dans des applications concrètes. Un grand merci également à l'équipe de Cajun-shop.com pour nous avoir permis d'utiliser leur site pour certains exemples de ce livre.

David tient à remercier sa famille, en particulier sa mère et son grand-père, pour leur soutien indéfectible. Il remercie également ceux qui ont supporté son emploi du temps capricieux au fil des années consacrées à l'écriture de ce livre. Il remercie également Slurp, Orctomi et Norma pour tout ce qu'ils ont fait, ainsi que ses collègues auteurs pour leur travail acharné.

Enfin, il aimerait remercier Brian de l’avoir entraîné dans une autre aventure.

Marjorie tient à remercier son mari, Alan Liu, pour ses conseils techniques, son soutien familial et sa compréhension. Elle remercie également ses collègues auteurs pour leurs nombreuses idées et inspirations. Elle est reconnaissante d'avoir pu collaborer ensemble à la rédaction de ce livre.

Sailu aimerait remercier David et Brian pour l'opportunité de travailler sur ce livre, et Chuck Neerdaels pour lui avoir fait découvrir HTTP.

Anshu tient à remercier sa femme, Rashi, et ses parents pour leur patience, leur soutien et leurs encouragements pendant les longues années passées à écrire ce livre.

Enfin, les auteurs remercient collectivement les pionniers d'Internet, célèbres et anonymes, dont les recherches, le développement et l'évangélisation au cours des quatre dernières décennies ont tant contribué à notre communauté scientifique, sociale et économique. Sans leur travail, ce livre n'aurait pas de sujet.

PARTIE I

I. HTTP : le fondement du Web

Cette section est une introduction au protocole HTTP. Les quatre chapitres suivants décrivent la technologie de base de HTTP, fondement du Web :

• Le chapitre 1, Présentation de HTTP, est un aperçu rapide de HTTP.

• Le chapitre 2, URL et ressources, détaille les formats des URL et les différents types de ressources qu'elles désignent sur Internet. Nous décrivons également l'évolution vers les URN.

• Chapitre 3, Messages HTTP, détaille les messages HTTP qui transportent le contenu Web.

• Le chapitre 4, Gestion des connexions, aborde les règles et comportements généralement mal compris et mal documentés pour la gestion des connexions TCP par HTTP.

Chapitre 1Ceci est le titre du livre CHAPITRE 1

Présentation de HTTP

Les navigateurs, serveurs et applications web du monde entier communiquent entre eux via HTTP (Hypertext Transfer Protocol). HTTP est le langage commun de l'Internet mondial moderne.

Ce chapitre offre un aperçu concis du protocole HTTP. Vous découvrirez comment les applications web utilisent HTTP pour communiquer et vous aurez une idée générale de son fonctionnement. Nous aborderons notamment :

• Comment les clients et les serveurs Web communiquent

• D’où proviennent les ressources (contenu Web)

• Comment fonctionnent les transactions Web

• Le format des messages utilisés pour la communication HTTP

• Le transport réseau TCP sous-jacent

• Les différentes variantes du protocole HTTP

• Certains des nombreux composants architecturaux HTTP installés sur InternetNous avons beaucoup de terrain à couvrir, alors commençons notre visite de HTTP.

HTTP : le courrier multimédia d'Internet

Des milliards d'images JPEG, de pages HTML, de fichiers texte, de films MPEG, de fichiers audio WAV, d'applets Java et bien plus encore circulent chaque jour sur Internet. HTTP transfère la majeure partie de ces informations de manière rapide, pratique et fiable depuis des serveurs web du monde entier vers les navigateurs web des utilisateurs.

Grâce à des protocoles de transmission de données fiables, HTTP garantit que vos données ne seront ni endommagées ni brouillées pendant leur transmission, même lorsqu'elles proviennent de l'autre bout du monde. C'est un avantage pour vous, utilisateur, car vous pouvez accéder aux informations sans vous soucier de leur intégrité. Une transmission fiable est également avantageuse pour vous, développeur d'applications Internet, car vous n'avez pas à vous soucier des communications HTTP.

être détruit, dupliqué ou déformé pendant le transport. Vous pouvez vous concentrer sur la programmation des détails distinctifs de votre application, sans vous soucier des failles et des faiblesses d'Internet.

Examinons de plus près comment HTTP transporte le trafic Web.

Clients et serveurs Web

Le contenu Web réside sur des serveurs Web. Ces derniers utilisent le protocole HTTP, d'où leur nom de serveurs HTTP. Ces serveurs stockent les données Internet et les fournissent à la demande des clients HTTP. Ces derniers envoient des requêtes HTTP aux serveurs, qui renvoient les données demandées dans des réponses HTTP, comme illustré à la figure 1-1. Ensemble, les clients et les serveurs HTTP constituent les composants de base du World Wide Web.

Figure 1-1. Clients et serveurs Web

Vous utilisez probablement des clients HTTP au quotidien. Le client le plus courant est un navigateur web, tel que Microsoft Internet Explorer ou Netscape Navigator. Les navigateurs web demandent des objets HTTP aux serveurs et les affichent à l'écran.

Lorsque vous accédez à une page, telle que « http://www.oreilly.com/index.html », votre navigateur envoie une requête HTTP au serveur www.oreilly.com (voir Figure 1-1). Le serveur tente de trouver l'objet recherché (ici, « /index.html ») et, en cas de succès, l'envoie au client dans une réponse HTTP, avec son type, sa longueur et d'autres informations.

Ressources

Les serveurs web hébergent des ressources web. Une ressource web est la source du contenu web. Le type de ressource web le plus simple est un fichier statique sur le système de fichiers du serveur web. Ces fichiers peuvent contenir n'importe quoi : des fichiers texte, HTML, Microsoft Word, Adobe Acrobat, des images JPEG, des films AVI ou tout autre format imaginable.

Cependant, les ressources ne sont pas nécessairement des fichiers statiques. Il peut également s'agir de logiciels générant du contenu à la demande. Ces ressources de contenu dynamique peuvent générer du contenu en fonction de votre identité, des informations que vous avez demandées ou de l'heure de la journée. Elles peuvent vous montrer une image en direct d'une caméra, vous permettre de négocier des actions, de consulter des bases de données immobilières ou d'acheter des cadeaux en ligne (voir figure 1-2).

Figure 1-2. Une ressource Web est tout ce qui fournit du contenu Web.

En résumé, une ressource est toute source de contenu. Un fichier contenant la feuille de calcul des prévisions de ventes de votre entreprise est une ressource. Une passerelle web permettant de consulter les rayons de votre bibliothèque publique locale est une ressource. Un moteur de recherche Internet est une ressource.

Types de médias

Étant donné qu'Internet héberge des milliers de types de données différents, HTTP étiquette soigneusement chaque objet transporté sur le Web avec une étiquette de format de données appelée type MIME. MIME (Multipurpose Internet Mail Extensions) a été initialement conçu pour résoudre les problèmes rencontrés lors du transfert de messages entre différents systèmes de messagerie électronique. MIME a si bien fonctionné pour le courrier électronique que HTTP l'a adopté pour décrire et étiqueter son propre contenu multimédia.

Les serveurs web associent un type MIME à toutes les données d'objet HTTP (voir Figure 1-3). Lorsqu'un navigateur web reçoit un objet d'un serveur, il examine le type MIME associé pour vérifier s'il sait comment le gérer. La plupart des navigateurs gèrent des centaines de types d'objets courants : affichage de fichiers image, analyse et formatage de fichiers HTML, lecture de fichiers audio via les haut-parleurs de l'ordinateur ou lancement de plug-ins externes pour gérer des formats spéciaux.

Ressources

Figure 1-3. Les types MIME sont renvoyés avec le contenu des données

Un type MIME est une étiquette textuelle, représentée par un type d'objet principal et un sous-type spécifique, séparés par une barre oblique. Par exemple :

• Un document texte au format HTML serait étiqueté avec le type text/html.

• Un document texte ASCII brut serait étiqueté avec le type texte/brut.

• Une version JPEG d'une image serait image/jpeg.

• Une image au format GIF serait image/gif.

• Un film Apple QuickTime serait une vidéo/quicktime.

• Une présentation Microsoft PowerPoint serait application/vnd.ms-powerpoint.

Il existe des centaines de types MIME courants, ainsi que de nombreux autres types expérimentaux ou à usage limité. Une liste très complète des types MIME est fournie en annexe D.

URI

Chaque ressource du serveur Web possède un nom, permettant aux clients d'indiquer les ressources qui les intéressent. Ce nom est appelé identifiant uniforme de ressource (URI). Les URI sont comparables aux adresses postales d'Internet : ils identifient et localisent de manière unique les ressources d'information dans le monde entier.

Voici un URI pour une ressource d'image sur le serveur Web du magasin Joe's Hardware :

http://www.joes-hardware.com/specials/saw-blade.gif

La figure 1-4 montre comment l'URI spécifie le protocole HTTP pour accéder à la ressource GIF de la lame de scie sur le serveur du magasin de Joe. L'URI étant donnée, HTTP peut récupérer l'objet. Il existe deux types d'URI : les URL et les URN. Examinons maintenant chacun de ces types d'identifiants de ressource.

URL

L'URL (Uniform Resource Locator) est la forme la plus courante d'identifiant de ressource. Les URL décrivent l'emplacement précis d'une ressource sur un serveur particulier. Elles indiquent précisément comment récupérer une ressource à partir d'un emplacement fixe et précis. La figure 1-4 montre comment une URL indique précisément où se trouve une ressource et comment y accéder. Le tableau 1-1 présente quelques exemples d'URL.

Figure 1-4. Les URL spécifient le protocole, le serveur et la ressource locale

Tableau 1-1. Exemples d'URL

Description de l'URL

http://www.oreilly.com/index.html L'URL d'accueil de O'Reilly & Associates, Inc.

http://www.yahoo.com/images/logo.gif L'URL du logo du site Web Yahoo!

http://www.joes-hardware.com/inventory-check.

cgi?item=12731 L'URL d'un programme qui vérifie si l'article d'inventaire

#12731 est en stock

ftp://joe: tools4u@ftp.joes-hardware.com /lockingpliers.gif L'URL du fichier image locking-pliers.gif, en utilisant le protocole d'accès FTP protégé par mot de passe

La plupart des URL suivent un format standardisé composé de trois parties principales :

• La première partie de l'URL est appelée « schéma » et décrit le protocole utilisé pour accéder à la ressource. Il s'agit généralement du protocole HTTP (http://).

• La deuxième partie donne l'adresse Internet du serveur (par exemple, www.joes-hardware.com).

• Le reste nomme une ressource sur le serveur Web (par exemple, /specials/saw-blade.gif).

Aujourd’hui, presque chaque URI est une URL.

URN

Le deuxième type d'URI est le nom uniforme de ressource, ou URN. Un URN sert de nom unique pour un contenu particulier, quel que soit l'emplacement de la ressource. Ces URN, indépendants de l'emplacement, permettent aux ressources de se déplacer d'un endroit à un autre. Ils permettent également l'accès aux ressources par plusieurs protocoles d'accès réseau tout en conservant le même nom.

Par exemple, l'URN suivante pourrait être utilisée pour nommer le document de normes Internet « RFC 2141 », quel que soit son emplacement (elle peut même être copiée dans plusieurs

lieux):

urne:ietf:rfc:2141

Ressources

Les URN sont encore expérimentaux et peu répandus. Pour fonctionner efficacement, ils nécessitent une infrastructure de support pour localiser les ressources ; l'absence d'une telle infrastructure a également ralenti leur adoption. Cependant, les URN offrent des perspectives prometteuses. Nous aborderons les URN plus en détail au chapitre 2, mais le reste de ce livre se concentre presque exclusivement sur les URL.

Sauf indication contraire, nous adoptons la terminologie conventionnelle et utilisons URI et URL de manière interchangeable pour le reste de ce livre.

Transactions

Examinons plus en détail comment les clients utilisent HTTP pour traiter avec les serveurs web et leurs ressources. Une transaction HTTP se compose d'une commande (envoyée du client au serveur) et d'une réponse (envoyée du serveur au client). Cette communication s'effectue via des blocs de données formatés appelés messages HTTP, comme illustré à la figure 1-5.

Figure 1-5. Les transactions HTTP se composent de messages de requête et de réponse

Méthodes

HTTP prend en charge plusieurs commandes de requête, appelées méthodes HTTP. Chaque requête HTTP possède une méthode. Cette méthode indique au serveur l'action à effectuer (récupérer une page web, exécuter une passerelle, supprimer un fichier, etc.). Le tableau 1-2 répertorie cinq méthodes HTTP courantes.

Tableau 1-2. Quelques méthodes HTTP courantes

Description de la méthode HTTP

GET Envoyer la ressource nommée du serveur au client.

PUT Stockez les données du client dans une ressource de serveur nommée.

Tableau 1-2. Quelques méthodes HTTP courantes (suite)

Description de la méthode HTTP

SUPPRIMER Supprime la ressource nommée d'un serveur.

POST Envoyer les données client dans une application de passerelle serveur.

HEAD Envoyez uniquement les en-têtes HTTP de la réponse pour la ressource nommée.

Nous discuterons des méthodes HTTP en détail dans le chapitre 3.

Codes d'état

Chaque message de réponse HTTP est accompagné d'un code d'état. Ce code numérique à trois chiffres indique au client si la requête a abouti ou si d'autres actions sont nécessaires. Le tableau 1-3 présente quelques codes d'état courants.

Tableau 1-3. Quelques codes d'état HTTP courants

Code d'état HTTP Description

200 OK. Document retourné correctement.

Redirection 302. Allez ailleurs pour obtenir la ressource.

404 Introuvable. Impossible de trouver cette ressource.

HTTP envoie également une phrase explicative textuelle avec chaque code d'état numérique (voir le message de réponse de la figure 1-5). La phrase textuelle est incluse uniquement à des fins descriptives ; le code numérique est utilisé pour tous les traitements.

Les codes d'état et les phrases de raison suivants sont traités de manière identique par le logiciel HTTP :

200 OK

200 Document joint

200 succès

200 Tout va bien, mec

Les codes d'état HTTP sont expliqués en détail au chapitre 3.

Les pages Web peuvent être constituées de plusieurs objets

Une application émet souvent plusieurs transactions HTTP pour accomplir une tâche. Par exemple, un navigateur web émet une cascade de transactions HTTP pour récupérer et afficher une page web riche en graphiques. Le navigateur effectue une transaction pour récupérer le squelette HTML décrivant la mise en page, puis émet des transactions HTTP supplémentaires pour chaque image intégrée, volet graphique, applet Java, etc. Ces ressources intégrées peuvent même résider sur des serveurs différents, comme illustré à la figure 1-6. Ainsi, un

« page Web » est souvent un ensemble de ressources, et non une ressource unique.

Transactions

Figure 1-6. Les pages Web composites nécessitent des transactions HTTP distinctes pour chaque ressource intégrée.

Messages

Examinons maintenant rapidement la structure des messages de requête et de réponse HTTP. Nous étudierons les messages HTTP en détail au chapitre 3.

Les messages HTTP sont des séquences de caractères simples, orientées ligne. Étant du texte brut et non binaire, ils sont faciles à lire et à écrire pour les humains.* La figure 1-7 illustre les messages HTTP d'une transaction simple.

(a) Message de demande (b) Message de réponse

HTTP/1.0 200 OK

Salut ! Je suis un message !

OBTENIR /test/hi-there.txt HTTP/1.0

Ligne de départ

En-têtes

Corps

Figure 1-7. Les messages HTTP ont une structure de texte simple, orientée ligne

Les messages HTTP envoyés par les clients web aux serveurs web sont appelés messages de requête. Les messages des serveurs aux clients sont appelés messages de réponse. Il n'existe pas d'autres types de messages HTTP. Les formats des messages de requête et de réponse HTTP sont très similaires.

* Certains programmeurs se plaignent de la difficulté d'analyse HTTP, qui peut être complexe et source d'erreurs, notamment lors de la conception de logiciels à haut débit. Un format binaire ou un format texte plus restreint aurait pu être plus simple à traiter, mais la plupart des programmeurs HTTP apprécient son extensibilité et sa déboguabilité.

Les messages HTTP se composent de trois parties :

Ligne de départ

La première ligne du message est la ligne de départ, indiquant ce qu'il faut faire pour une demande ou ce qui s'est passé pour une réponse.

Champs d'en-tête

La ligne de départ est suivie de zéro ou plusieurs champs d'en-tête. Chaque champ d'en-tête se compose d'un nom et d'une valeur, séparés par deux points (:) pour faciliter l'analyse. Les en-têtes se terminent par une ligne vide. Ajouter un champ d'en-tête est aussi simple que d'ajouter une ligne.

Corps

Après la ligne vide se trouve un corps de message facultatif contenant tout type de données. Les corps de requête transmettent les données au serveur web ; les corps de réponse les renvoient au client. Contrairement aux lignes de départ et aux en-têtes, qui sont textuels et structurés, le corps peut contenir des données binaires arbitraires (par exemple, des images, des vidéos, des pistes audio, des applications logicielles). Bien entendu, le corps peut également contenir du texte.

Exemple de message simple

La figure 1-8 montre les messages HTTP pouvant être envoyés dans le cadre d'une transaction simple. Le navigateur demande la ressource http://www.joes-hardware.com/tools.html.

Dans la figure 1-8, le navigateur envoie un message de requête HTTP. La requête comporte une méthode GET dans sa ligne de départ et la ressource locale est /tools.html. La requête indique qu'elle utilise la version 1.0 du protocole HTTP. Le message de requête est vide, car aucune donnée n'est nécessaire pour obtenir un document simple auprès d'un serveur.

Le serveur renvoie un message de réponse HTTP. Cette réponse contient le numéro de version HTTP (HTTP/1.0), un code de réussite (200), une phrase de justification descriptive (OK) et un bloc de champs d'en-tête de réponse, le tout suivi du corps de la réponse contenant le document demandé. La longueur du corps de la réponse est indiquée dans l'en-tête ContentLength et le type MIME du document dans Content-Type.

en-tête.

Relations

Maintenant que nous avons esquissé à quoi ressemblent les messages HTTP, parlons un instant de la façon dont les messages se déplacent d'un endroit à un autre, via les connexions TCP (Transmission Control Protocol).

TCP/IP

HTTP est un protocole de couche applicative. Il ne se préoccupe pas des détails de la communication réseau ; il laisse la gestion du réseau à TCP/IP, le protocole de transport Internet fiable et populaire.

Figure 1-8. Exemple de transaction GET pour http://www.joes-hardware.com/tools.html

TCP fournit :

• Transport de données sans erreur

• Livraison dans l'ordre (les données arriveront toujours dans l'ordre dans lequel elles ont été envoyées)

• Flux de données non segmenté (peut diffuser des données de n'importe quelle taille à tout moment)

Internet repose sur TCP/IP, un ensemble de protocoles réseau à commutation de paquets en couches, répandu et utilisé par les ordinateurs et les périphériques réseau du monde entier. TCP/IP masque les particularités et les faiblesses des réseaux et matériels individuels, permettant ainsi aux ordinateurs et aux réseaux de tout type de communiquer entre eux de manière fiable.

Une fois la connexion TCP établie, les messages échangés entre les ordinateurs client et serveur ne seront jamais perdus, endommagés ou reçus dans le désordre.

En termes de réseau, le protocole HTTP est superposé à TCP. HTTP utilise TCP pour transporter ses données de message. De même, TCP est superposé à IP (voir figure 1-9).

Figure 1-9. Pile de protocoles réseau HTTP

Connexions, adresses IP et numéros de port

Avant qu'un client HTTP puisse envoyer un message à un serveur, il doit établir une connexion TCP/IP entre le client et le serveur en utilisant des adresses de protocole Internet (IP) et

numéros de port.

Établir une connexion TCP, c'est un peu comme appeler quelqu'un au siège social d'une entreprise. Vous composez d'abord le numéro de téléphone de l'entreprise. Vous êtes ainsi mis en relation avec la bonne organisation. Ensuite, vous composez le numéro de poste de la personne que vous souhaitez joindre.

Dans TCP, vous avez besoin de l'adresse IP de l'ordinateur serveur et du numéro de port TCP associé au programme logiciel spécifique exécuté sur le serveur.

C'est bien beau, mais comment obtenir l'adresse IP et le numéro de port du serveur HTTP ? L'URL, bien sûr ! Nous avons déjà mentionné que les URL sont les adresses des ressources. Elles peuvent donc naturellement nous fournir l'adresse IP de la machine qui les héberge. Examinons quelques URL :

http://207.200.83.29:80/index.html
http://www.netscape.com:80/index.html
http://www.netscape.com/index.html

La première URL contient l'adresse IP de la machine, « 207.200.83.29 », et le numéro de port, « 80 ».

La deuxième URL n'a pas d'adresse IP numérique ; elle possède un nom de domaine textuel, ou nom d'hôte (« www.netscape.com »). Le nom d'hôte est simplement un alias convivial pour une adresse IP. Les noms d'hôtes peuvent facilement être convertis en adresses IP grâce au service DNS (Domain Name Service). Nous sommes donc prêts pour cette étape. Nous reviendrons plus en détail sur le DNS et les URL au chapitre 2.

L'URL finale ne comporte pas de numéro de port. Lorsque le numéro de port est absent d'une requête HTTP,

URL, vous pouvez supposer la valeur par défaut du port 80.

Grâce à l'adresse IP et au numéro de port, un client peut facilement communiquer via TCP/IP. La figure 1-10 montre comment un navigateur utilise HTTP pour afficher une ressource HTML simple résidant sur un serveur distant.

Voici les étapes :

(a) Le navigateur extrait le nom d'hôte du serveur à partir de l'URL.

(b) Le navigateur convertit le nom d'hôte du serveur en adresse IP du serveur.

(c) Le navigateur extrait le numéro de port (le cas échéant) de l'URL.

(d) Le navigateur établit une connexion TCP avec le serveur Web.

(e) Le navigateur envoie un message de requête HTTP au serveur.

(f) Le serveur renvoie une réponse HTTP au navigateur.

(g) La connexion est fermée et le navigateur affiche le document.

Figure 1-10. Processus de connexion de base au navigateur

Un exemple réel utilisant Telnet

Étant donné que HTTP utilise TCP/IP et est basé sur du texte, au lieu d'utiliser un format binaire obscur, il est simple de communiquer directement avec un serveur Web.

L'utilitaire Telnet connecte votre clavier à un port TCP de destination et relie la sortie de ce port à votre écran. Telnet est couramment utilisé pour les sessions de terminal distant, mais il peut généralement se connecter à n'importe quel serveur TCP, y compris

Serveurs HTTP.

Vous pouvez utiliser l'utilitaire Telnet pour communiquer directement avec les serveurs web. Telnet vous permet d'ouvrir une connexion TCP sur un port d'une machine et d'y saisir des caractères directement. Le serveur web vous traite comme un client web et toutes les données renvoyées via la connexion TCP s'affichent à l'écran.

Utilisons Telnet pour interagir avec un véritable serveur web. Nous allons utiliser Telnet pour récupérer le document pointé par l'URL http://www.joes-hardware.com:80/tools.html (vous pouvez essayer cet exemple vous-même).

Passons en revue ce qui devrait se passer :

• Tout d'abord, nous devons rechercher l'adresse IP de www.joes-hardware.com et ouvrir un

Connexion TCP au port 80 de cette machine. Telnet effectue ce travail pour nous.

• Une fois la connexion TCP ouverte, nous devons saisir la requête HTTP.

• Lorsque la requête est terminée (indiquée par une ligne vide), le serveur doit renvoyer le contenu dans une réponse HTTP et fermer la connexion.

Notre exemple de requête HTTP pour http://www.joes-hardware.com:80/tools.html est présenté dans l'exemple 1-1. Le texte saisi est en gras.

Exemple 1-1. Une transaction HTTP utilisant Telnet

% telnet www.joes-hardware.com 80 Essai 161.58.228.45...

Connecté à joes-hardware.com.

Le caractère d'échappement est « ^] ».

OBTENIR /tools.html HTTP/1.1 Hôte : www.joes-hardware.com

HTTP/1.1 200 OK

Date : dim. 1er oct. 2000 23:25:17 GMT

Serveur : Apache/1.3.11 BSafe-SSL/1.38 (Unix) FrontPage/4.0.4.3

Dernière modification : mar. 4 juil. 2000 09:46:21 GMT

ETag : « 373979-193-3961b26d »

Accept-Ranges : octets

Contenu-Longueur : 403

Connexion : fermer

Type de contenu : texte/html

Exemple 1-1. Transaction HTTP via Telnet (suite)

<HTML>

<HEAD><TITLE>Les outils de Joe</TITLE></HEAD>

<CORPS>

<H1>Page Outils</H1>

<H2>Marteaux</H2>

<P>Joe's Hardware Online propose la plus grande sélection de marteaux au monde.</P>

<H2><A NAME=drills></A>Exercices</H2>

<P>Joe's Hardware propose une gamme complète de perceuses sans fil et avec fil, ainsi que les dernières perceuses atomiques alimentées au plutonium, pour les gros travaux autour de la maison.</P> ...

</BODY>

</HTML> Connexion fermée par un hôte étranger.

Telnet recherche le nom d'hôte et ouvre une connexion au serveur Web www.joes-hardware.com, qui écoute sur le port 80. Les trois lignes après la commande sont générées par Telnet, nous indiquant qu'il a établi une connexion.

Nous saisissons ensuite notre commande de requête de base, « GET /tools.html HTTP/1.1 », et envoyons un en-tête Host contenant le nom d'hôte d'origine, suivi d'une ligne vide, demandant au serveur de récupérer la ressource « /tools.html » depuis le serveur www.joes-hardware.com. Le serveur répond ensuite avec une ligne de réponse, plusieurs en-têtes de réponse, une ligne vide, puis le corps du document HTML.

Attention, Telnet imite bien les clients HTTP, mais ne fonctionne pas bien comme serveur. De plus, l'automatisation des scripts Telnet n'est pas du tout agréable. Pour un outil plus flexible, vous pouvez essayer nc (netcat). Cet outil vous permet de manipuler et de scripter facilement le trafic UDP et TCP, y compris HTTP. Consultez http://netcat.sourceforge.net pour plus de détails.

Versions du protocole

Plusieurs versions du protocole HTTP sont actuellement utilisées. Les applications HTTP doivent déployer des efforts considérables pour gérer efficacement les différentes variantes du protocole. Les versions actuellement utilisées sont :

HTTP/0.9

Le prototype de HTTP de 1991 est connu sous le nom de HTTP/0.9. Ce protocole présente de nombreux défauts de conception importants et ne doit être utilisé que pour interagir avec des clients existants. HTTP/0.9 ne prend en charge que la méthode GET et ne prend pas en charge le typage MIME du contenu multimédia, les en-têtes HTTP ni les numéros de version. HTTP/0.9 était initialement défini pour récupérer des objets HTML simples. Il a rapidement été remplacé par HTTP/1.0.

HTTP/1.0

La version 1.0 a été la première version de HTTP à être largement déployée. Elle a ajouté des numéros de version, des en-têtes HTTP, des méthodes supplémentaires et la gestion des objets multimédias. Elle a facilité la prise en charge de sites web graphiquement attrayants.

Pages et formulaires interactifs, qui ont contribué à l'adoption à grande échelle du World Wide Web. Cette spécification n'a jamais été clairement définie. Elle représentait un recueil de bonnes pratiques à une époque où le protocole évoluait rapidement, tant sur le plan commercial que académique.

HTTP/1.0+

Au milieu des années 1990, de nombreux clients et serveurs web populaires ont rapidement ajouté des fonctionnalités à HTTP afin de répondre aux exigences d'un Web en pleine expansion et au succès commercial. Nombre de ces fonctionnalités, notamment les connexions « keepalive » durables, la prise en charge de l'hébergement virtuel et des connexions proxy, ont été ajoutées à HTTP et sont devenues des normes non officielles, de facto. Cette version informelle et étendue de HTTP est souvent appelée HTTP/1.0+.

HTTP/1.1

HTTP/1.1 s'est concentré sur la correction des défauts architecturaux de HTTP, la spécification de la sémantique, l'introduction d'optimisations significatives des performances et la suppression des dysfonctionnements. HTTP/1.1 incluait également la prise en charge des applications web et des déploiements plus sophistiqués, en cours à la fin des années 1990.

HTTP/1.1 est la version actuelle de HTTP.

HTTP-NG (également appelé HTTP/2.0)

HTTP-NG est un prototype de proposition architecturale pour le successeur de HTTP/1.1. Il se concentre sur des optimisations significatives des performances et un framework plus puissant pour l'exécution à distance de la logique serveur. Les recherches sur HTTP-NG se sont achevées en 1998 et, au moment de la rédaction de ce

document, il n'est pas prévu de proposer cette proposition comme remplaçant de HTTP/1.1. Voir le chapitre 10 pour plus d'informations.

Composants architecturaux du Web

Dans ce chapitre de présentation, nous nous sommes concentrés sur la manière dont deux applications web (navigateurs et serveurs web) échangent des messages pour mettre en œuvre des transactions de base. Il existe de nombreuses autres applications web avec lesquelles vous interagissez sur Internet. Dans cette section, nous présenterons plusieurs autres applications importantes, notamment :

Procurations

Intermédiaires HTTP qui se situent entre les clients et les serveurs

Caches

Entrepôts HTTP qui conservent des copies de pages Web populaires à proximité des clients

Passerelles

Serveurs Web spéciaux qui se connectent à d'autres applications

Tunnels

Proxys spéciaux qui transmettent aveuglément les communications HTTP

Agents

Clients Web semi-intelligents qui effectuent des requêtes HTTP automatisées

Composants architecturaux du Web

Procurations

Commençons par examiner les serveurs proxy HTTP, éléments de base importants pour la sécurité Web, l’intégration des applications et l’optimisation des performances.

Comme le montre la figure 1-11, un proxy se situe entre un client et un serveur. Il reçoit toutes les requêtes HTTP du client et les relaie au serveur (éventuellement après les avoir modifiées). Ces applications agissent comme proxy pour l'utilisateur, accédant au serveur en son nom.

Figure 1-11. Les proxys relaient le trafic entre le client et le serveur.

Les proxys sont souvent utilisés à des fins de sécurité, agissant comme des intermédiaires de confiance par lesquels transite tout le trafic web. Ils peuvent également filtrer les requêtes et les réponses, par exemple

pour détecter les virus d'applications dans les téléchargements d'entreprise ou pour filtrer le contenu réservé aux adultes et éloigner les élèves du primaire. Nous aborderons les proxys en détail au chapitre 6.

Caches

Un cache Web, ou proxy de mise en cache, est un type particulier de serveur proxy HTTP qui conserve des copies des documents courants transitant par le proxy. Le client suivant demandant le même document peut être servi à partir de la copie personnelle du cache (voir figure 1-12).

Figure 1-12. Les proxys de mise en cache conservent des copies locales des documents populaires pour améliorer les performances.

Un client peut télécharger un document beaucoup plus rapidement depuis un cache proche que depuis un serveur web distant. HTTP définit de nombreuses fonctionnalités pour optimiser la mise en cache et réguler la fraîcheur et la confidentialité du contenu mis en cache. Nous abordons la technologie de mise en cache au chapitre 7.

Passerelles

Les passerelles sont des serveurs spécifiques qui servent d'intermédiaires pour d'autres serveurs. Elles sont souvent utilisées pour convertir le trafic HTTP vers un autre protocole. Une passerelle reçoit toujours les requêtes comme si elle était le serveur d'origine de la ressource. Le client peut ignorer qu'il communique avec une passerelle.

Par exemple, une passerelle HTTP/FTP reçoit des requêtes d'URI FTP via des requêtes HTTP, mais récupère les documents via le protocole FTP (voir figure 1-13). Le document résultant est compressé dans un message HTTP et envoyé au client. Nous aborderons les passerelles au chapitre 8.

Figure 1-13. Passerelle HTTP/FTP

Tunnels

Les tunnels sont des applications HTTP qui, après configuration, relaient aveuglément des données brutes entre deux connexions. Les tunnels HTTP sont souvent utilisés pour transporter des données non HTTP sur une ou plusieurs connexions HTTP, sans consulter les données.

Une utilisation courante des tunnels HTTP consiste à acheminer le trafic SSL (Secure Sockets Layer) chiffré via une connexion HTTP, autorisant ainsi le trafic SSL à traverser les pare-feu d'entreprise qui

n'autorisent que le trafic web. Comme illustré à la figure 1-14, un tunnel HTTP/SSL reçoit une requête HTTP pour établir une connexion sortante vers une adresse et un port de destination, puis achemine le trafic SSL chiffré via le canal HTTP afin de le relayer en aveugle vers le serveur de destination.

Agents

Les agents utilisateurs (ou simplement agents) sont des programmes clients qui effectuent des requêtes HTTP pour le compte de l'utilisateur. Toute application émettant des requêtes web est un agent HTTP. Jusqu'à présent, nous n'avons abordé qu'un seul type d'agent HTTP : les navigateurs web. Mais il existe de nombreux autres types d'agents utilisateurs.

Composants architecturaux du Web

Figure 1-14. Les tunnels transmettent les données sur des réseaux non HTTP (tunnel HTTP/SSL illustré)

Par exemple, il existe des agents utilisateurs automatisés qui parcourent le Web de manière autonome, émettant des transactions HTTP et récupérant du contenu, sans supervision humaine. Ces agents automatisés portent souvent des noms évocateurs, tels que « araignées » ou « robots web » (voir figure 1-15). Les araignées parcourent le Web pour constituer des archives de contenu utiles, comme la base de données d'un moteur de recherche ou un catalogue de produits pour un robot comparateur de prix. Voir le chapitre 9 pour plus d'informations.

Figure 1-15. Les robots d'indexation des moteurs de recherche automatisés sont des agents qui récupèrent des pages web partout dans le monde.

La fin du commencement

Voilà pour notre brève introduction à HTTP. Dans ce chapitre, nous avons mis en avant le rôle de HTTP comme protocole de transport multimédia. Nous avons expliqué comment HTTP utilise les URI pour nommer les ressources multimédias sur des serveurs distants, comment les messages de requête et de réponse HTTP sont utilisés pour manipuler les ressources multimédias sur des serveurs distants, et nous avons terminé par une présentation de quelques applications web utilisant HTTP.

Les chapitres restants expliquent de manière beaucoup plus détaillée les mécanismes techniques du protocole HTTP, les applications et les ressources.

Pour plus d'informations

Les chapitres suivants de ce livre exploreront HTTP de manière beaucoup plus détaillée, mais vous constaterez peut-être que certaines des sources suivantes contiennent des informations utiles sur des sujets particuliers que nous avons abordés dans ce chapitre.

Informations sur le protocole HTTP

Référence HTTP Pocket

Ce petit livre fournit une introduction concise à HTTP et une référence rapide à chacun des en-têtes et codes d'état qui composent les transactions HTTP.

http://www.w3.org/Protocols/

Cette page Web du W3C contient de nombreux liens utiles sur le protocole HTTP.

http://www.ietf.org/rfc/rfc2616.txt

La RFC 2616, « Hypertext Transfer Protocol—HTTP/1.1 », est la spécification officielle de HTTP/1.1, la version actuelle du protocole HTTP. Bien rédigée, bien structurée et détaillée, cette spécification n'est toutefois pas idéale pour les lecteurs souhaitant comprendre les concepts et les motivations sous-jacents de HTTP, ni les différences entre théorie et pratique. Nous espérons que ce livre vous éclairera sur les concepts sous-jacents afin que vous puissiez mieux exploiter la spécification.

http://www.ietf.org/rfc/rfc1945.txt

La RFC 1945, « Hypertext Transfer Protocol—HTTP/1.0 », est une RFC informative qui décrit les fondements modernes du protocole HTTP. Elle détaille le comportement officiel et les bonnes pratiques des applications web au moment de la rédaction de la spécification. Elle contient également des descriptions utiles sur des comportements obsolètes dans HTTP/1.1, mais encore largement implémentés par les applications traditionnelles.

http://www.w3.org/Protocols/HTTP/AsImplemented.html

Cette page Web contient une description du protocole HTTP/0.9 de 1991, qui implémente uniquement les requêtes GET et n'a aucun typage de contenu.

Pour plus d'informations

Perspective historique

http://www.w3.org/Protocols/WhyHTTP.html

Cette brève page Web de 1991, rédigée par l'auteur de HTTP, met en évidence certains des objectifs originaux et minimalistes de HTTP.

http://www.w3.org/History.html

« Une petite histoire du World Wide Web » donne une perspective courte mais intéressante sur certains des premiers objectifs et fondements du World Wide Web et du HTTP.

http://www.w3.org/DesignIssues/Architecture.html

« Web Architecture from 50,000 Feet » dresse un portrait large et ambitieux du World Wide Web et des principes de conception qui affectent HTTP et les sites Web associés.

technologies.

Autres informations sur le World Wide Web

http://www.w3.org

Le World Wide Web Consortium (W3C) est l'équipe de pilotage technologique du Web. Le W3C développe des technologies interopérables (spécifications, directives, logiciels et outils) pour un Web en constante évolution. Le site du W3C est une mine d'or de documentations introductives et détaillées sur les technologies Web.

http://www.ietf.org/rfc/rfc2396.txt

La RFC 2396, « Uniform Resource Identifiers (URI) : syntaxe générique », est la référence détaillée pour les URI et les URL.

http://www.ietf.org/rfc/rfc2141.txt

La RFC 2141, « Syntaxe URN », est une spécification de 1997 décrivant la syntaxe URN.

http://www.ietf.org/rfc/rfc2046.txt

La RFC 2046, « MIME Partie 2 : Types de médias », est la deuxième d'une série de cinq spécifications Internet définissant la norme Multipurpose Internet Mail Extensions pour la gestion de contenu multimédia.

http://www.wrec.org/Drafts/draft-ietf-wrec-taxonomy-06.txt

Ce projet Internet, « Internet Web Replication and Caching Taxonomy », spécifie la terminologie standard pour les composants architecturaux Web.

Chapitre 2Ceci est le titre du livre CHAPITRE 2

URL et ressources

Imaginez Internet comme une ville géante en pleine expansion, regorgeant de lieux à voir et d'activités. Vous, les autres résidents et

touristes de cette communauté en plein essor, utiliseriez des conventions de nommage standard pour les nombreuses attractions et services de la ville. Vous utiliseriez des adresses pour les musées, les restaurants et les domiciles. Vous utiliseriez des numéros de téléphone pour les pompiers, la secrétaire du patron et votre mère, qui vous reproche de ne pas appeler assez souvent.

Tout a un nom standardisé, pour faciliter l'organisation des ressources de la ville. Les livres ont un numéro ISBN, les bus ont un numéro de ligne, les comptes bancaires ont un numéro de compte et les gens ont un numéro de sécurité sociale. Demain, vous retrouverez vos partenaires commerciaux à la porte 31 de l'aéroport. Chaque matin, vous prenez une ligne rouge et descendez à la station Kendall Square.

Et comme tout le monde s'est mis d'accord sur des normes pour ces différents noms, nous pouvons facilement partager les trésors de la ville. Vous pouvez demander au chauffeur de taxi de vous emmener au 246 McAllister Street, et il comprendra ce que vous voulez dire (même s'il prend le chemin le plus long).

Les localisateurs de ressources uniformes (URL) sont les noms normalisés des ressources Internet. Les URL pointent vers des informations électroniques, indiquant leur emplacement et comment interagir avec elles.

Dans ce chapitre, nous aborderons :

• Syntaxe de l'URL et signification et fonction des différents composants de l'URL

• Raccourcis URL pris en charge par de nombreux clients Web, notamment les URL relatives et les URL expandomatiques

• Codage des URL et règles de caractères

• Schémas d'URL courants qui prennent en charge une variété de systèmes d'information Internet

• L'avenir des URL, y compris les noms de ressources uniformes (URN) : un cadre pour prendre en charge les objets qui se déplacent d'un endroit à un autre tout en conservant des noms stables

Naviguer dans les ressources Internet

Les URL sont les emplacements de ressources dont votre navigateur a besoin pour trouver des informations. Elles permettent aux utilisateurs et aux applications de trouver, d'utiliser et de partager les milliards de données disponibles sur Internet. Les URL sont le point d'accès humain habituel au protocole HTTP et aux autres protocoles : un utilisateur pointe un navigateur vers une URL et, en coulisses, le navigateur envoie les messages de protocole appropriés pour obtenir la ressource souhaitée.

Les URL sont en réalité un sous-ensemble d'une classe plus générale d'identifiants de ressources appelée identifiant uniforme de ressource (URI). Les URI sont un concept général composé de deux sous-ensembles principaux : les URL et les URN. Les URL identifient les ressources en décrivant leur emplacement, tandis que les URN (que nous aborderons plus loin dans ce chapitre) identifient les ressources par leur nom, quel que soit leur emplacement actuel.

La spécification HTTP utilise le concept plus général d'URI comme identifiant de ressources ; en pratique, cependant, les applications HTTP ne traitent que le sous-ensemble URL des URI. Tout au long de ce livre, nous utiliserons parfois les termes URI et URL de manière interchangeable, mais nous parlerons presque toujours d'URL.

Supposons que vous souhaitiez récupérer l'URL http://www.joes-hardware.com/seasonal/index-fall.html :

• La première partie de l'URL (http) constitue le schéma d'URL. Ce schéma indique au client web comment accéder à la ressource. Dans ce cas, l'URL indique d'utiliser le protocole HTTP.

• La deuxième partie de l'URL (www.joes-hardware.com) est l'emplacement du serveur.

Cela indique au client Web où la ressource est hébergée.

• La troisième partie de l'URL (/seasonal/index-fall.html) correspond au chemin d'accès à la ressource. Ce chemin indique la ressource locale demandée sur le serveur.

Voir la figure 2-1 pour une illustration.

Figure 2-1. Liens entre les URL et le navigateur, la machine, le serveur et l'emplacement sur le système de fichiers du serveur.

Les URL peuvent vous diriger vers des ressources disponibles via des protocoles autres que HTTP.

Ils peuvent vous orienter vers n'importe quelle ressource sur Internet, à partir du compte de messagerie d'une personne :

mailto: president@whitehouse.gov

aux fichiers disponibles via d'autres protocoles, tels que le protocole de transfert de fichiers (FTP) :

ftp://ftp.lots-o-books.com/pub/complete-price-list.xls

aux films hébergés sur des serveurs de streaming vidéo :

rtsp://www.joes-hardware.com:554/interview/cto_video

Les URL permettent de nommer uniformément les ressources. La plupart des URL ont le même nom.

Structure « scheme://server location/path ». Ainsi, pour chaque ressource et chaque moyen d'y accéder, il existe une seule façon de nommer chaque ressource afin que chacun puisse l'utiliser pour la trouver. Cependant, cela n'a pas toujours été le cas.

Les jours sombres avant les URL

Avant le Web et les URL, les utilisateurs s'appuyaient sur un assortiment hétéroclite d'applications pour accéder aux données réparties sur le Net. La plupart d'entre eux n'avaient pas la chance de disposer de toutes les applications nécessaires ou manquaient de savoir-faire et de patience pour les utiliser.

Avant l'apparition des URL, pour partager le fichier complete-catalog.xls avec un ami, il fallait dire quelque chose comme : « Utilisez FTP pour vous connecter à ftp.joes-hardware.com. Connectez-vous en mode anonyme. Saisissez ensuite votre nom d'utilisateur comme mot de passe. Accédez au répertoire pub. Passez en mode binaire. Téléchargez ensuite le fichier complete-catalog.xls sur votre système de fichiers local et visualisez-le. »

Aujourd'hui, des navigateurs comme Netscape Navigator et Microsoft Internet Explorer regroupent une grande partie de ces fonctionnalités dans un package pratique. Grâce aux URL, ces applications peuvent accéder à de nombreuses ressources de manière uniforme, via une interface unique. Au lieu des instructions compliquées ci-dessus, vous pourriez simplement dire : « Pointez votre

navigateur à ftp://ftp.lots-o-books.com/pub/complete-catalog.xls.”

Les URL permettent aux applications de savoir comment accéder à une ressource. En réalité, de nombreux utilisateurs ignorent probablement les protocoles et méthodes d'accès utilisés par leur navigateur pour obtenir les ressources demandées.

Grâce aux navigateurs web, plus besoin de lecteur de news pour consulter les actualités sur Internet ni de client FTP pour accéder aux fichiers sur des serveurs FTP. Plus besoin de messagerie électronique pour envoyer et recevoir des e-mails. Les URL ont contribué à simplifier le monde en ligne, en permettant au navigateur d'accéder et de gérer intelligemment les ressources.*

Les applications peuvent utiliser des URL pour simplifier l’accès aux informations.

Les URL vous fournissent, à vous et à votre navigateur, tout ce dont vous avez besoin pour trouver une information. Elles définissent la ressource recherchée, son emplacement et comment l'obtenir.

* Les navigateurs utilisent souvent d'autres applications pour gérer des ressources spécifiques. Par exemple, Internet Explorer lance une

application de messagerie pour gérer les URL qui identifient les ressources de messagerie.

Naviguer dans les ressources Internet

Syntaxe de l'URL

Les URL fournissent un moyen de localiser n'importe quelle ressource sur Internet, mais ces ressources sont accessibles par différents schémas (par exemple, HTTP, FTP, SMTP) et la syntaxe des URL varie d'un schéma à l'autre.

Cela signifie-t-il que chaque schéma d'URL possède une syntaxe radicalement différente ? En pratique, non. La plupart des URL adhèrent à une syntaxe générale, et il existe un chevauchement important de style et de syntaxe entre les différents schémas d'URL.

La plupart des schémas d'URL basent leur syntaxe d'URL sur ce format général en neuf parties :

<schéma>://<utilisateur>:<mot de passe>@<hôte>:<port>/<chemin>;<params>?<query>#<frag>

Presque aucune URL ne contient tous ces éléments. Les trois éléments les plus importants d'une URL sont le schéma, l'hôte et le chemin. Le tableau 2-1 récapitule les différents composants.

Tableau 2-1. Composants généraux de l'URL

Description du composant Valeur par défaut

Schéma Quel protocole utiliser pour accéder à un serveur pour obtenir une ressource. Aucun

utilisateur Le nom d'utilisateur que certains schémas nécessitent pour accéder à une ressource.

mot de passe Le mot de passe qui peut être inclus après le nom d'utilisateur, séparé par deux points (:). <Adresse e-mail>

hôte Le nom d'hôte ou l'adresse IP en pointillés du serveur hébergeant la ressource. Aucun

Port : le numéro de port utilisé par le serveur hébergeant la ressource. De nombreux protocoles ont des numéros de port par défaut (le numéro de port par défaut pour HTTP est 80). Spécifique au protocole.

Chemin : nom local de la ressource sur le serveur, séparé des composants de l'URL précédents par une barre oblique (/). La syntaxe du composant « chemin » est spécifique au serveur et au schéma. (Nous verrons plus loin dans ce chapitre que le chemin d'une URL peut être divisé en segments, et que chaque segment peut avoir ses propres composants spécifiques.) Aucun.

paramètres : utilisés par certains schémas pour spécifier les paramètres d'entrée. Les paramètres sont des paires nom/valeur. Une URL peut contenir plusieurs champs de paramètres, séparés d'eux-mêmes et du reste du chemin par des points-virgules (;). Aucun.

Requête : utilisée par certains schémas pour transmettre des paramètres à des applications actives (bases de données, forums, moteurs de recherche et autres passerelles Internet). Il n'existe pas de format commun pour le contenu de la requête. Elle est séparée du reste de l'URL par le caractère « ? ». Aucun

frag : nom d'une partie de la ressource. Le champ frag n'est pas transmis au serveur lors du référencement de l'objet ; il est utilisé en interne par le client. Il est séparé du reste de l'URL par le caractère « # ». Aucun.

Prenons par exemple l'URL http://www.joes-hardware.com:80/index.html. Le schéma est « http », l'hôte est « www.joes-hardware.com », le port est « 80 » et le chemin est « /index.html ».

Schémas : quel protocole utiliser

Le schéma est en réalité l'identifiant principal permettant d'accéder à une ressource donnée ; il indique à l'application qui interprète l'URL le protocole à utiliser. Dans notre URL HTTP simple, le schéma est simplement « http ».

Le composant du schéma doit commencer par un caractère alphabétique et être séparé du reste de l'URL par le premier caractère « : ». Les noms de schémas ne sont pas sensibles à la casse ; les URL « http://www.joes-hardware.com » et « HTTP://www.joeshardware.com » sont donc équivalentes.

Hôtes et ports

Pour trouver une ressource sur Internet, une application doit savoir quelle machine héberge cette ressource et où trouver le serveur ayant accès à la ressource souhaitée. Les composants hôte et port de l'URL fournissent ces deux informations.

Le composant hôte identifie la machine hôte sur Internet qui a accès à la ressource. Le nom peut être un nom d'hôte, comme indiqué ci-dessus (« www.joes-hardware.com »), ou une adresse IP. Par exemple, les deux URL suivantes pointent vers la même ressource : la première désigne le serveur par son nom d'hôte et la seconde par son adresse IP :

http://www.joes-hardware.com:80/index.html
http://161.58.228.45:80/index.html

Le composant port identifie le port réseau sur lequel le serveur écoute. Pour HTTP, qui utilise le protocole TCP sous-jacent, le port par défaut est 80.

Noms d'utilisateur et mots de passe

Les composants utilisateur et mot de passe sont plus intéressants. De nombreux serveurs nécessitent un nom d'utilisateur et un mot de passe pour accéder aux données. Les serveurs FTP en sont un exemple courant. En voici quelques exemples :

ftp://ftp.prep.ai.mit.edu/pub/gnu ftp: //anonymous@ftp.prep.ai.mit.edu /pub/gnu ftp://anonymous: my_passwd@ftp.prep.ai.mit.edu /pub/gnu http://joe: joespasswd@www.joes-hardware.com /sales_info.txt

Le premier exemple ne contient ni utilisateur ni mot de passe, mais uniquement notre schéma standard, l'hôte et le chemin d'accès. Si une application utilise un schéma d'URL nécessitant un nom d'utilisateur et un mot de passe, comme FTP, elle insère généralement un nom d'utilisateur et un mot de passe par défaut s'ils ne sont pas fournis. Par exemple, si vous fournissez à votre navigateur une URL FTP sans spécifier de nom d'utilisateur ni de mot de passe, il insèrera « anonymous » comme nom d'utilisateur et enverra un mot de passe par défaut (Internet Explorer envoie « IEUser », tandis que Netscape Navigator envoie « mozilla »).

Syntaxe de l'URL

Le deuxième exemple montre un nom d'utilisateur spécifié comme « anonyme ». Ce nom d'utilisateur, combiné au composant hôte, ressemble à une adresse e-mail. Le caractère « @ » sépare les composants utilisateur et mot de passe du reste de l'URL.

Dans le troisième exemple, un nom d'utilisateur (« anonymous ») et un mot de passe (« my_passwd ») sont spécifiés, séparés par le caractère « : ».

Chemins

Le chemin d'accès de l'URL indique l'emplacement de la ressource sur le serveur. Ce chemin ressemble souvent à un chemin d'accès hiérarchique dans un système de fichiers. Par exemple :

http://www.joes-hardware.com:80/seasonal/index-fall.html

Le chemin d'accès de cette URL est « /seasonal/index-fall.html », ce qui ressemble à un chemin d'accès de système de fichiers Unix. Ce chemin contient les informations nécessaires au serveur pour localiser la ressource.* Le chemin d'accès des URL HTTP peut être divisé en segments séparés par des caractères « / » (comme dans un chemin d'accès de fichier Unix). Chaque segment peut avoir son propre composant « paramètres ».

Paramètres

Pour de nombreux schémas, un simple hôte et un chemin d'accès à l'objet ne suffisent pas. Outre le port d'écoute du serveur et l'accès à la ressource avec un nom d'utilisateur et un mot de passe, de nombreux protocoles nécessitent davantage d'informations pour fonctionner.

Les applications interprétant les URL ont besoin de ces paramètres de protocole pour accéder à la ressource. Sinon, le serveur distant risque de ne pas traiter la requête, voire de la traiter incorrectement. Prenons l'exemple d'un protocole comme FTP, qui propose deux modes de transfert : binaire et texte. Il est déconseillé de transférer l'image binaire en mode texte, car elle pourrait être brouillée.

Pour fournir aux applications les paramètres d'entrée nécessaires à une communication correcte avec le serveur, les URL disposent d'un composant « params ». Ce composant est simplement une liste de paires nom/valeur dans l'URL, séparées du reste de l'URL (et entre elles) par des caractères « ; ». Il fournit aux applications toutes les informations supplémentaires nécessaires pour accéder à la ressource. Par exemple :

ftp://prep.ai.mit.edu/pub/gnu;type=d

Dans cet exemple, il y a un paramètre, type=d, où le nom du paramètre est « type » et sa valeur est « d ».

* Ceci est une simplification. Dans la section « Hébergement virtuel » du chapitre 18, nous verrons que le chemin d'accès ne suffit pas toujours à localiser une ressource. Parfois, un serveur a besoin d'informations supplémentaires.

Comme mentionné précédemment, le chemin d'accès des URL HTTP peut être divisé en segments. Chaque segment peut avoir ses propres paramètres. Par exemple :

http://www.joes-hardware.com/hammers;sale=false/index.html;graphics=true

Dans cet exemple, il y a deux segments de chemin : hammers et index.html. Le segment hammers contient le paramètre « sale » et sa valeur est « false ». Le segment index.html contient le paramètre « graphics » et sa valeur est « true ».

Chaînes de requête

Certaines ressources, telles que les services de base de données, peuvent faire l'objet de questions ou de requêtes pour affiner le type de ressource demandée.

Imaginons que la quincaillerie Joe's conserve une liste de ses invendus dans une base de données et permette d'interroger ce stock pour savoir si les produits sont en stock. L'URL suivante pourrait être utilisée

pour interroger une passerelle de base de données web afin de vérifier si l'article numéro 12731 est disponible :

http://www.joes-hardware.com/inventory-check.cgi?item=12731

Dans l'ensemble, cela ressemble aux autres URL que nous avons étudiées. La nouveauté réside dans tout ce qui se trouve à droite du point d'interrogation (?). C'est ce qu'on appelle le composant de requête. Ce composant de requête de l'URL est transmis à une ressource de passerelle, le composant de chemin de l'URL identifiant cette ressource. En résumé, les passerelles peuvent être considérées comme des points d'accès à d'autres applications (nous les détaillons au chapitre 8).

La figure 2-2 montre un exemple de composant de requête transmis à un serveur servant de passerelle vers l'application de vérification d'inventaire de Joe's Hardware. La requête vérifie si un article particulier, 12731, est en stock en taille et en couleur.

bleu.

Figure 2-2. Le composant de requête URL est envoyé à l'application passerelle.

Le format du composant de requête n'est pas obligatoire, sauf que certains caractères sont illégaux, comme nous le verrons plus loin dans ce chapitre. Par convention, de nombreuses passerelles

Syntaxe de l'URL

attendez-vous à ce que la chaîne de requête soit formatée comme une série de paires « nom=valeur », séparées par des caractères « & » :

http://www.joes-hardware.com/inventory-check.cgi?item=12731&color=blue

Dans cet exemple, il y a deux paires nom/valeur dans le composant de requête : item=12731 et color=blue.

Fragments

Certains types de ressources, comme le HTML, peuvent être subdivisés au-delà du simple niveau de ressource. Par exemple, pour un seul document texte volumineux comportant des sections, l'URL de la ressource pointe vers l'intégralité du document texte, mais idéalement, vous pouvez spécifier les sections de la ressource.

Pour permettre le référencement de parties ou de fragments d'une ressource, les URL prennent en charge un composant frag permettant d'identifier les éléments d'une ressource. Par exemple, une URL peut pointer vers une image ou une section spécifique d'un document HTML.

Un fragment est suspendu à droite d'une URL, précédé du caractère #. Par exemple :

http://www.joes-hardware.com/tools.html#drills

Dans cet exemple, le fragment drills fait référence à une partie de la page web /tools.html située sur le serveur web de Joe's Hardware. Cette partie est nommée « drills ».

Étant donné que les serveurs HTTP ne traitent généralement que des objets entiers* et non des fragments d'objets, les clients ne transmettent pas de fragments aux serveurs (voir figure 2-3). Une fois que votre navigateur a récupéré la ressource entière du serveur, il utilise le fragment pour afficher la partie de la ressource qui vous intéresse.

Raccourcis URL

Les clients web comprennent et utilisent quelques raccourcis URL. Les URL relatives sont un raccourci pratique pour spécifier une ressource au sein d'une autre ressource. De nombreux navigateurs prennent également en charge l'extension automatique des URL : l'utilisateur peut saisir une partie clé (mémorisable) d'une URL, et le navigateur complète le reste. Ceci est expliqué dans la section « URL expandomatiques ».

URL relatives

Les URL existent en deux types : absolues et relatives. Jusqu'ici, nous n'avons étudié que les URL absolues. Avec une URL absolue, vous disposez de toutes les informations nécessaires pour accéder à une page.

ressource.

* Dans la section « Requêtes de plage » du chapitre 15, nous verrons que les agents HTTP peuvent demander des plages d'octets d'objets. Cependant, dans le cas de fragments d'URL, le serveur envoie l'objet entier et l'agent applique l'identifiant du fragment à la ressource.

Figure 2-3. Le fragment d'URL est utilisé uniquement par le client, car le serveur gère des objets entiers.

En revanche, les URL relatives sont incomplètes. Pour obtenir toutes les informations nécessaires à l'accès à une ressource à partir d'une URL relative, vous devez l'interpréter par rapport à une autre URL, appelée sa base.

Les URL relatives constituent une abréviation pratique pour les URL. Si vous avez déjà écrit du code HTML à la main, vous les avez probablement trouvées très utiles.

L'exemple 2-1 contient un exemple de document HTML avec une URL relative intégrée.

Exemple 2-1. Extrait HTML avec URL relatives

<HTML>

<HEAD><TITLE>Les outils de Joe</TITLE></HEAD>

<CORPS>

<H1> Page Outils </H1>

<H2> Marteaux <H2>

<P>Joe's Hardware Online propose la plus grande sélection de <A HREF="./hammers.html">marteaux </A> au monde.

</BODY>

</HTML>

Dans l'exemple 2-1, nous avons un document HTML pour la ressource :

http://www.joes-hardware.com/tools.html

Dans le document HTML, il existe un lien hypertexte contenant l'URL ./hammers.html. Cette URL semble incomplète, mais il s'agit d'une URL relative légale. Elle peut être interprétée par rapport à l'URL du document dans lequel elle se trouve ; dans ce cas, par rapport à la ressource /tools.html sur le serveur web de Joe's Hardware.

Raccourcis URL

La syntaxe abrégée des URL relatives permet aux auteurs HTML d'omettre le schéma, l'hôte et d'autres composants des URL. Ces composants peuvent être déduits de l'URL de base de la ressource dans laquelle ils se trouvent. Les URL d'autres ressources peuvent également être spécifiées avec cette abréviation.

Dans l'exemple 2-1, notre URL de base est :

http://www.joes-hardware.com/tools.html

En utilisant cette URL comme base, nous pouvons déduire les informations manquantes. Nous savons que la ressource est ./hammers.html, mais nous ignorons le schéma ni l'hôte. En utilisant l'URL de base, nous pouvons déduire que le schéma est http et que l'hôte est www.joes-hardware.com.

La figure 2-4 illustre cela.

Figure 2-4. Utilisation d'une URL de base

Les URL relatives ne sont que des fragments ou des parties d'URL. Les applications qui traitent les URL (comme votre navigateur) doivent pouvoir convertir les URL relatives en URL absolues.

Il est également intéressant de noter que les URL relatives constituent un moyen pratique de garantir la portabilité d'un ensemble de ressources (comme des pages HTML). L'utilisation d'URL relatives permet de déplacer un ensemble de documents sans que leurs liens ne soient interrompus, car ils seront interprétés par rapport à la nouvelle base. Cela permet notamment de dupliquer du contenu sur d'autres documents.

serveurs.

URL de base

La première étape du processus de conversion consiste à trouver une URL de base. Cette URL sert de point de référence pour l'URL relative. Elle peut provenir de plusieurs sources :

Fourni explicitement dans la ressource

Certaines ressources spécifient explicitement l'URL de base. Un document HTML, par exemple, peut inclure une balise HTML <BASE> définissant l'URL de base à utiliser pour convertir toutes les URL relatives de ce document HTML.

URL de base de la ressource encapsulante

Si une URL relative est trouvée dans une ressource qui ne spécifie pas explicitement une URL de base, comme dans l'exemple 2-1, elle peut utiliser l'URL de la ressource dans laquelle elle est intégrée comme base (comme nous l'avons fait dans notre exemple).

Aucune URL de base

Dans certains cas, il n'existe pas d'URL de base. Cela signifie souvent que vous disposez d'une URL absolue ; cependant, il peut arriver que vous ayez simplement une URL incomplète ou rompue.

Résolution des références relatives

Nous avons précédemment présenté les composants de base et la syntaxe des URL. L'étape suivante pour convertir une URL relative en URL absolue consiste à décomposer les URL relatives et de base en leurs éléments constitutifs.

En réalité, vous analysez simplement l'URL, mais cette opération est souvent appelée décomposition, car vous la décomposez en ses composants. Une fois les URL de base et relatives décomposées, vous pouvez appliquer l'algorithme illustré à la figure 2-5 pour finaliser la conversion.

Figure 2-5. Conversion d'URL relatives en URL absolues

Raccourcis URL

Cet algorithme convertit une URL relative en sa forme absolue, qui peut ensuite être utilisée pour référencer la ressource. Cet algorithme a été initialement spécifié dans la RFC 1808, puis intégré à la RFC 2396.

Avec notre exemple ./hammers.html de l'exemple 2-1, nous pouvons appliquer l'algorithme décrit dans la figure 2-5 :

1. Le chemin est ./hammers.html ; l'URL de base est http://www.joes-hardware.com/tools.html.

2. Le schéma est vide ; continuez vers le bas de la moitié gauche du graphique et héritez du schéma d'URL de base (HTTP).

3. Au moins un composant n'est pas vide ; passez au bas de la page en héritant des composants hôte et port.

4. En combinant les composants que nous avons de l'URL relative (chemin : ./hammers.html) avec ce que nous avons hérité (schéma : http, hôte : www.joes-hardware.com, port : 80), nous obtenons notre nouvelle URL absolue : http://www.joes-hardware.com/hammers.html.

URL Expandomatic

Certains navigateurs tentent de développer automatiquement les URL, soit après leur soumission, soit pendant leur saisie. Cela offre un raccourci aux utilisateurs : ils n'ont pas besoin de saisir l'URL complète, car elle se développe automatiquement.

Ces fonctionnalités « expandomatiques » sont disponibles en deux versions :

Extension du nom d'hôte

Lors de l'extension du nom d'hôte, le navigateur peut souvent étendre le nom d'hôte que vous saisissez au nom d'hôte complet sans votre aide, simplement en utilisant quelques heuristiques simples.

Par exemple, si vous saisissez « yahoo » dans la barre d'adresse, votre navigateur peut automatiquement insérer « www. » et « .com » dans le nom d'hôte, créant ainsi « www.yahoo.com ». Certains navigateurs tentent cette méthode s'ils ne trouvent pas de site correspondant à « yahoo », en essayant plusieurs extensions avant d'abandonner. Ces astuces simples vous permettent de gagner du temps et d'éviter les frustrations.

Cependant, ces astuces d'extension des noms d'hôtes peuvent poser problème à d'autres applications HTTP, comme les proxys. Nous aborderons ces problèmes plus en détail au chapitre 6.

Extension de l'histoire

Une autre technique utilisée par les navigateurs pour vous faire gagner du temps lors de la saisie des URL consiste à enregistrer l'historique des URL consultées. À mesure que vous saisissez l'URL, ils vous proposent des choix complets en comparant votre saisie aux préfixes des URL de votre historique. Ainsi, si vous saisissez le début d'une URL précédemment consultée, comme http://www.joes-, votre navigateur pourrait suggérer http://www.joes-hardware.com. Vous pourriez alors sélectionner cette option au lieu de saisir l'URL complète.

Veuillez noter que l'extension automatique des URL peut se comporter différemment avec les proxys. Ce point est abordé plus en détail dans la section « Extension automatique des clients URI et résolution du nom d'hôte » du chapitre 6.

Personnages louches

Les URL ont été conçues pour être portables. Elles ont également été conçues pour nommer uniformément toutes les ressources sur Internet, ce qui signifie qu'elles seront transmises via différents protocoles. Comme ces protocoles utilisent des mécanismes différents pour transmettre leurs données, il était important que les URL soient conçues pour pouvoir être transmises en toute sécurité via n'importe quel protocole Internet.

Une transmission sécurisée signifie que les URL peuvent être transmises sans risque de perte d'informations. Certains protocoles, comme le protocole SMTP (Simple Mail Transfer Protocol) pour le courrier électronique, utilisent des méthodes de transmission permettant de supprimer certains caractères.* Pour contourner ce problème, les URL ne peuvent contenir que des caractères d'un alphabet relativement petit et universellement sûr.

Outre la portabilité des URL par tous les protocoles Internet, les concepteurs souhaitaient qu'elles soient lisibles par tous. Les caractères invisibles et non imprimables sont donc également interdits dans les URL, même s'ils peuvent être transmis par les services de messagerie et être portables.†

Pour compliquer encore les choses, les URL doivent également être complètes. Les concepteurs d'URL ont compris que certaines personnes souhaiteraient que leurs URL contiennent des données binaires ou des caractères hors de l'alphabet universel. Un mécanisme d'échappement a donc été ajouté, permettant de coder des caractères non sécurisés en caractères sécurisés pour le transport.

Cette section résume l’alphabet universel et les règles de codage des URL.

Le jeu de caractères de l'URL

Les jeux de caractères par défaut des systèmes informatiques présentent souvent un biais anglocentrique. Historiquement, de

nombreuses applications informatiques ont utilisé le jeu de caractères US-ASCII. Ce dernier utilise 7 bits pour représenter la plupart des touches disponibles sur une machine à écrire anglaise, ainsi que quelques caractères de contrôle non imprimables pour le formatage du texte et la signalisation matérielle.

L'US-ASCII est très portable, grâce à sa longue tradition. Cependant, bien qu'il soit pratique pour les citoyens américains, il ne prend pas en charge les caractères fléchis courants dans les langues européennes ni les centaines de langues non romanes lues par des milliards de personnes dans le monde.

* Ceci est dû à l'utilisation d'un codage 7 bits pour les messages ; cela peut supprimer des informations si la source est codée en 8 bits ou plus.

† Les caractères non imprimables incluent les espaces (notez que la RFC 2396 recommande aux applications d'ignorer les espaces).

Personnages louches

De plus, certaines URL peuvent contenir des données binaires arbitraires. Conscients de ce besoin d'exhaustivité, les concepteurs d'URL ont intégré des séquences d'échappement. Ces séquences permettent de coder des valeurs de caractères ou des données arbitraires à l'aide d'un sous-ensemble restreint du jeu de caractères US-ASCII, garantissant ainsi portabilité et exhaustivité.

Mécanismes d'encodage

Pour contourner les limitations d'une représentation sécurisée des jeux de caractères, un schéma de codage a été conçu pour représenter les caractères non sécurisés d'une URL. Ce codage représente simplement le caractère non sécurisé par une notation d'échappement, composée d'un signe de pourcentage (%) suivi de deux chiffres hexadécimaux représentant le code ASCII du caractère.

Le tableau 2-2 montre quelques exemples.

Tableau 2-2. Exemples de caractères codés

Exemple d'URL de code ASCII de caractères

~ 126 (0x7E) http://www.joes-hardware.com/%7Ejoe

ESPACE 32 (0x20) http://www.joes-hardware.com/more%20tools.html

% 37 (0x25) http://www.joes-hardware.com/100%25satisfaction.html

Restrictions de caractères

Plusieurs caractères ont été réservés pour une signification particulière dans une URL. D'autres ne font pas partie du jeu imprimable US-ASCII défini. D'autres encore sont connus pour perturber certaines

passerelles et protocoles Internet ; leur utilisation est donc déconseillée.

Le tableau 2-3 répertorie les caractères qui doivent être codés dans une URL avant de les utiliser à d'autres fins que celles qui leur sont réservées.

Tableau 2-3. Caractères réservés et restreints

Réservation/restriction de caractère

% Réservé comme jeton d'échappement pour les caractères codés

/ Réservé à la délimitation et à la division des segments de chemin dans le composant de chemin

. Réservé dans le composant de chemin

.. Réservé dans le composant de chemin

Réservé comme délimiteur de fragment

? Réservé comme délimiteur de chaîne de requête

; Réservé comme délimiteur de paramètres

: Réservé pour délimiter les composants schéma, utilisateur/mot de passe et hôte/port

$, + Réservé

@ & = Réservés car ils ont une signification particulière dans le contexte de certains schémas

Tableau 2-3. Caractères réservés et restreints (suite)

Réservation/restriction de caractère

{ } | \ ^ ~ [ ] ' Restreint en raison d'une manipulation dangereuse par divers agents de transport, tels que les passerelles

< > " Non sécurisé ; doit être codé car ces caractères ont souvent une signification en dehors de la portée de l'URL, comme la délimitation de l'URL elle-même dans un document (par exemple, « http://www.joes-hardware.com »)

0x00–0x1F, 0x7F Restreint ; les caractères dans ces plages hexadécimales appartiennent à la section non imprimable du jeu de caractères US-ASCII

> 0x7F Restreint ; les caractères dont les valeurs hexadécimales se situent dans cette plage ne se situent pas dans la plage de 7 bits du système de codage américain.

jeu de caractères ASCII

Un peu plus

Vous vous demandez peut-être pourquoi rien de grave ne s'est produit alors que vous avez utilisé des caractères dangereux. Par exemple, vous pouvez consulter la page d'accueil de Joe à l'adresse suivante :

http://www.joes-hardware.com/~joe

et ne pas encoder le caractère « ~ ». Pour certains protocoles de transport, ce n'est pas un problème, mais il est déconseillé aux développeurs d'applications d'encoder les caractères non sécurisés.

Les applications doivent trouver le juste milieu. Il est préférable que les applications clientes convertissent les caractères non sécurisés ou restreints avant d'envoyer une URL à une autre application.* Une fois tous les caractères non sécurisés encodés, l'URL est sous une forme canonique et peut être partagée entre les applications ; il n'y a donc pas lieu de craindre que l'autre application soit perturbée par la signification particulière des caractères.

L'application d'origine qui reçoit l'URL de l'utilisateur est la mieux placée pour déterminer les caractères à encoder. Chaque composant de l'URL pouvant contenir ses propres caractères, et ces caractères étant définis selon le schéma, seule l'application recevant l'URL de l'utilisateur est réellement en mesure de déterminer les caractères à encoder.

Bien sûr, l'autre extrême consiste à ce que l'application encode tous les caractères. Bien que cela ne soit pas recommandé, il n'existe aucune règle absolue interdisant l'encodage de caractères déjà considérés comme sûrs ; cependant, en pratique, cela peut entraîner des comportements étranges et erronés, car certaines applications peuvent supposer que les caractères sûrs ne seront pas encodés.

Parfois, des personnes malveillantes encodent des caractères supplémentaires pour contourner les applications qui effectuent des correspondances de motifs sur les URL, par exemple les applications de filtrage web. Le codage de composants d'URL sécurisés peut empêcher les applications de correspondance de motifs de reconnaître les motifs recherchés. En général, les applications qui interprètent les URL doivent les décoder avant de les traiter.

* Nous parlons ici spécifiquement des applications clientes, et non d'autres intermédiaires HTTP, comme les proxys. Dans la section « Modification d'URI en cours » du chapitre 6, nous abordons certains problèmes pouvant survenir lorsque des proxys ou d'autres applications HTTP intermédiaires tentent de modifier (par exemple, encoder) des URL pour le compte d'un client.

Personnages louches

Certains composants d'URL, comme le schéma, doivent être facilement reconnaissables et doivent commencer par un caractère alphabétique. Consultez la section « Syntaxe des URL » pour plus d'informations sur

l'utilisation des caractères réservés et non sécurisés dans les différents composants d'URL.*

Une mer de plans

Dans cette section, nous examinerons les formats de schémas les plus courants sur le Web. L'annexe A fournit une liste assez exhaustive des schémas et des références à leur documentation.

Le tableau 2-4 résume certains des schémas les plus courants. Consulter la section « Syntaxe des URL » vous permettra de vous familiariser avec la syntaxe du tableau.

Tableau 2-4. Formats de schémas courants

Description du schéma

Le protocole http est conforme au format général des URL, à l'exception de l'absence de nom d'utilisateur et de mot de passe. Le port par défaut est 80 s'il est omis.

Forme de base : http://<hôte>:<port>/<chemin>?<requête>#<frag>

Exemples :

http://www.joes-hardware.com/index.html http://www.joes-hardware.com:80/index.html

https : Le protocole https est un jumeau du protocole http. La seule différence est qu'il utilise le protocole SSL (Secure Sockets Layer) de Netscape, qui assure le chiffrement de bout en bout des connexions HTTP. Sa syntaxe est identique à celle de HTTP, avec un port par défaut de 443.

Forme de base : https://<hôte>:<port>/<chemin>?<requête>#<frag> Exemple :

https://www.joes-hardware.com/secure.html

Les URL Mailto font référence à des adresses e-mail. Le comportement de l'e-mail étant différent des autres schémas (il ne fait pas référence à

(objets accessibles directement), le format d'une URL mailto diffère de celui de l'URL standard. La syntaxe des adresses e-mail Internet est documentée dans la RFC 822.

Forme de base : mailto :<RFC-822-addr-spec> Exemple : mailto : joe@joes-hardware.com

* Le tableau 2-3 répertorie les caractères réservés aux différents composants d'URL. En général, le codage doit être limité aux caractères non sécurisés pour le transport.

Tableau 2-4. Formats de schémas courants (suite)

Description du schéma

Les URL du protocole de transfert de fichiers ftp peuvent être utilisées pour télécharger et charger des fichiers sur un serveur FTP et pour obtenir des listes du contenu d'une structure de répertoire sur un serveur FTP.

Le protocole FTP existait avant l'avènement du Web et des URL. Les applications Web l'ont assimilé comme système d'accès aux données. La syntaxe des URL suit la forme générale.

Forme de base : ftp://<utilisateur>:<mot de passe>@<hôte>:<port>/<chemin>;<params> Exemple :

ftp://anonymous:joe% 40joes-hardware.com@prep.ai.mit.edu :21/pub/gnu/

Les URL RTSP sont des identifiants pour les ressources multimédias audio et vidéo qui peuvent être récupérées via le protocole de streaming en temps réel.

Le « u » dans le schéma rtspu indique que le protocole UDP est utilisé pour récupérer la ressource.

Formes de base :

rtsp://<utilisateur>:<mot de passe>@<hôte>:<port>/<chemin>
rtspu://<utilisateur>:<mot de passe>@<hôte>:<port>/<chemin>
Exemple :

rtsp://www.joes-hardware.com:554/interview/cto_video

Le schéma de fichier désigne les fichiers directement accessibles sur une machine hôte donnée (par un disque local, un système de fichiers réseau ou un autre système de partage de fichiers). Les champs suivent la forme générale. Si l'hôte est omis, la valeur par défaut est l'hôte local à partir duquel l'URL est utilisée.

Forme de base : file://<hôte>/<chemin> Exemple :

fichier://OFFICE-FS/policies/casual-fridays.doc

Le schéma d'actualités est utilisé pour accéder à des articles ou groupes de discussion spécifiques, tels que définis par la RFC 1036. Il a la propriété inhabituelle qu'une URL d'actualités en elle-même ne contient pas suffisamment d'informations pour localiser la ressource.

L'URL d'actualité ne contient aucune information sur l'emplacement d'acquisition de la ressource : aucun nom d'hôte ni de machine n'est fourni. C'est à l'application d'interprétation d'obtenir ces informations auprès de l'utilisateur. Par exemple, dans votre navigateur Netscape, dans le menu Options, vous pouvez spécifier votre serveur NNTP (actualités). Cela indique à votre navigateur quel serveur utiliser lorsqu'il dispose d'une URL d'actualités.

Les ressources d'actualités sont accessibles depuis plusieurs serveurs. Elles sont dites « indépendantes de la localisation », car elles ne dépendent d'aucune source d'accès.

Le caractère « @ » est réservé dans une URL d'actualité et est utilisé pour distinguer les URL d'actualité qui font référence à des groupes de discussion et les URL d'actualité qui font référence à des articles d'actualité spécifiques.

Formes de base :

news:<newsgroup> news:<news-article-id> Exemple :

actualités:rec.arts.startrek

Une mer de plans

Tableau 2-4. Formats de schémas courants (suite)

Description du schéma

Le schéma telnet est utilisé pour accéder à des services interactifs. Il ne représente pas un objet en soi, mais une application interactive (ressource) accessible via le protocole telnet.

Forme de base : telnet://<utilisateur>:<mot de passe>@<hôte>:<port>/ Exemple :

telnet://slurp: webhound@joes-hardware.com :23/

L'avenir

Les URL sont un outil puissant. Leur conception leur permet de nommer tous les objets existants et d'intégrer facilement de nouveaux formats. Elles offrent un mécanisme de nommage uniforme et partageable entre les protocoles Internet.

Cependant, elles ne sont pas parfaites. Les URL sont en réalité des adresses, et non de véritables noms. Cela signifie qu'une URL indique l'emplacement d'un objet, pour un instant donné. Elle fournit le nom d'un serveur spécifique sur un port spécifique, où trouver la ressource. L'inconvénient de ce système est que si la ressource est déplacée, l'URL n'est plus valide. Et à ce stade, elle ne permet plus de localiser l'objet.

L'idéal serait de connaître le vrai nom d'un objet, afin de pouvoir le retrouver où qu'il se trouve. Comme pour une personne, avec le nom de la ressource et quelques informations complémentaires, vous pourriez la localiser, où qu'elle se trouve.

L'Internet Engineering Task Force (IETF) travaille depuis un certain temps sur une nouvelle norme, les noms uniformes de ressources (URN), pour répondre précisément à ce problème. Les URN fournissent un nom stable à un objet, quel que soit son emplacement.

(soit à l'intérieur d'un serveur Web, soit entre plusieurs serveurs Web).

Les localisateurs uniformes de ressources (PURL) persistants illustrent la fonctionnalité URN grâce aux URL. Le concept consiste à introduire un niveau d'indirection supplémentaire dans la recherche d'une ressource, en utilisant un serveur localisateur intermédiaire qui catalogue et suit l'URL réelle de la ressource. Un client peut demander une URL persistante au localisateur, qui peut alors répondre avec une ressource redirigeant le client vers l'URL réelle et actuelle de la ressource (voir Figure 2-6). Pour plus d'informations sur les PURL, consultez le site http://purl.oclc.org.

Si ce n’est pas maintenant, quand ?

Les concepts à l'origine des URN existent depuis un certain temps. En effet, si l'on examine les dates de publication de certaines de leurs spécifications, on peut se demander pourquoi elles n'ont pas encore été adoptées.

Figure 2-6. Les PURL utilisent un serveur de localisation de ressources pour nommer l'emplacement actuel d'une ressource.

Passer des URL aux URN est une tâche colossale. La normalisation est un processus lent, souvent pour de bonnes raisons. La prise en charge des URN nécessitera de nombreux changements : consensus des organismes de normalisation, modifications de diverses applications HTTP, etc. Une masse critique considérable est nécessaire pour réaliser de tels changements, et malheureusement (ou peut-être heureusement), l'élan autour des URL est tel qu'il faudra du temps avant que tous les astres ne s'alignent pour rendre une telle conversion possible.

Tout au long de la croissance fulgurante du Web, les internautes – des informaticiens aux internautes lambda – ont appris à utiliser les URL. Bien qu'elles souffrent d'une syntaxe maladroite (pour les novices) et de problèmes de persistance, elles ont appris à les utiliser et à gérer leurs inconvénients. Les URL présentent certaines limites, mais elles ne constituent pas le problème le plus urgent pour la communauté des développeurs web.

Actuellement, et pour un avenir prévisible, les URL constituent le moyen de nommer les ressources sur Internet. Elles sont omniprésentes et ont joué un rôle essentiel dans le succès du Web. Il faudra du temps avant qu'un autre système de nommage ne supplante les URL. Cependant, les URL ont leurs limites, et il est probable que de nouvelles normes (éventuellement des URN) émergeront et seront déployées pour pallier certaines de ces limitations.

Pour plus d'informations

Pour plus d'informations sur les URL, reportez-vous à :

http://www.w3.org/Addressing/

La page Web du W3C sur la dénomination et l'adressage des URI et des URL.

http://www.ietf.org/rfc/rfc1738

RFC 1738, « Localisateurs de ressources uniformes (URL) », par T. Berners-Lee, L. Masinter et M. McCahill.

Pour plus d'informations

http://www.ietf.org/rfc/rfc2396.txt

RFC 2396, « Identificateurs de ressources uniformes (URI) : syntaxe générique », par T. BernersLee, R. Fielding et L. Masinter.

http://www.ietf.org/rfc/rfc2141.txt

RFC 2141, « Syntaxe URN », par R. Moats. http://purl.oclc.org

Le site Web du localisateur uniforme de ressources persistant.

http://www.ietf.org/rfc/rfc1808.txt

RFC 1808, « Localisateurs de ressources uniformes relatifs », par R. Fielding.

Chapitre 3l Ceci est le titre du livre CHAPITRE 3

Messages HTTP

Si HTTP est le coursier d'Internet, les messages HTTP sont les paquets qu'il utilise pour déplacer des données. Au chapitre 1, nous avons montré comment les programmes HTTP s'échangent des messages pour effectuer leur travail. Ce chapitre vous explique tout sur les messages HTTP : comment les créer et les comprendre. Après avoir lu ce chapitre, vous maîtriserez l'essentiel du nécessaire pour développer vos propres applications HTTP. Vous comprendrez notamment :

• Comment les messages circulent

• Les trois parties des messages HTTP (ligne de départ, en-têtes et corps de l'entité)

• Les différences entre les messages de demande et de réponse

• Les différentes fonctions (méthodes) qui demandent des messages prennent en charge

• Les différents codes d’état renvoyés avec les messages de réponse

• À quoi servent les différents en-têtes HTTP

Le flux des messages

Les messages HTTP sont des blocs de données échangés entre applications HTTP. Ces blocs de données commencent par des méta-informations textuelles décrivant le contenu et la signification du message, suivies de données facultatives. Ces messages circulent entre clients, serveurs et proxys. Les termes « entrant », « sortant », « amont » et « aval » décrivent la direction du message.

Les messages sont envoyés vers le serveur d'origine

HTTP utilise les termes entrant et sortant pour décrire le sens des transactions. Les messages transitent vers le serveur d'origine et, une fois leur tâche terminée, retournent vers l'agent utilisateur (voir figure 3-1).

Figure 3-1. Les messages transitent vers le serveur d'origine et reviennent vers le client.

Les messages circulent en aval

Les messages HTTP circulent comme des rivières. Tous les messages circulent en aval, qu'il s'agisse de requêtes ou de réponses (voir figure 3-2). L'expéditeur de tout message se trouve en amont du récepteur. Dans la figure 3-2, le proxy 1 est en amont du proxy 3 pour la requête, mais en aval du proxy 3 pour la réponse.*

Les parties d'un message

Les messages HTTP sont des blocs de données simples et formatés. La figure 3-3 en est un exemple. Chaque message contient soit une requête d'un client, soit une réponse d'un serveur. Ils se composent de trois parties : une ligne de départ décrivant le message, un bloc d'en-têtes contenant des attributs et un corps de message facultatif contenant des données.

La ligne de début et les en-têtes sont simplement du texte ASCII, divisé en lignes. Chaque ligne se termine par une séquence de fin de ligne de deux caractères, composée d'un retour chariot (ASCII 13) et d'un saut de ligne (ASCII 10). Cette séquence de fin de ligne s'écrit « CRLF ». Il est important de noter que, bien que la spécification HTTP pour la fin des lignes soit CRLF, les applications robustes devraient également accepter un simple saut de ligne. Certaines applications HTTP anciennes ou défectueuses n'envoient pas toujours le retour chariot et le saut de ligne.

Le corps de l'entité, ou corps du message (ou simplement « corps »), est simplement un bloc de données facultatif. Contrairement à la ligne de départ et aux en-têtes, le corps peut contenir du texte, des données binaires ou être vide.

Dans l'exemple de la figure 3-3, les en-têtes fournissent quelques informations sur le corps du message. La ligne « Content-Type » indique la nature du corps ; dans cet exemple, il s'agit d'un document texte brut. La ligne « Content-Length » indique sa taille ; ici, elle ne dépasse pas 19 octets.

* Les termes « en amont » et « en aval » ne concernent que l'expéditeur et le destinataire. Il est impossible de savoir si un message est destiné au serveur d'origine ou au client, car les deux sont en aval.

Syntaxe du message

Tous les messages HTTP se divisent en deux types : les messages de requête et les messages de réponse. Les messages de requête sollicitent une action auprès d'un serveur web. Les messages de réponse transmettent les résultats d'une requête au client. Les messages de requête et de réponse ont la même structure de base. La figure 3-4 illustre les messages de requête et de réponse pour obtenir une image GIF.

Voici le format d'un message de demande :

<méthode> <URL de requête> <version>

<en-têtes>

<entité-corps>

Figure 3-4. Une transaction HTTP comporte des messages de requête et de réponse.

Voici le format d'un message de réponse (notez que la syntaxe ne diffère que dans la ligne de départ) :

<version> <statut> <phrase-raison>

<en-têtes>

<entité-corps>

Voici une brève description des différentes parties :

méthode

Action que le client souhaite que le serveur exécute sur la ressource. Il s'agit d'un mot unique, comme « GET », « HEAD » ou « POST ». Nous détaillerons cette méthode plus loin dans ce chapitre.

URL de requête

Une URL complète nommant la ressource demandée, ou le chemin d'accès de l'URL. Si vous vous adressez directement au serveur, le

chemin d'accès de l'URL est généralement acceptable, à condition qu'il s'agisse du chemin absolu vers la ressource ; le serveur peut se considérer comme l'hôte/port de l'URL. Le chapitre 2 détaille la syntaxe des URL.

version

Version HTTP utilisée par le message. Son format est le suivant :

HTTP/<majeur>.<mineur>

où les valeurs majeure et mineure sont des entiers. Nous aborderons le contrôle de version HTTP plus en détail plus loin dans ce chapitre.

code d'état

Un nombre à trois chiffres décrivant le déroulement de la requête. Le premier chiffre de chaque code décrit la classe générale d'état (« succès », « erreur », etc.). Une liste exhaustive des codes d'état définis dans la spécification HTTP et de leur signification est fournie plus loin dans ce chapitre.

phrase de raison

Une version lisible du code d'état numérique, composée de tout le texte jusqu'à la séquence de fin de ligne. Des exemples de phrases de raison pour tous les codes d'état définis dans la spécification HTTP sont fournis plus loin dans ce chapitre. La phrase de raison est destinée uniquement à une utilisation humaine ; par exemple, les lignes de réponse contenant « HTTP/1.0 200 NOT OK » et « HTTP/1.0 200 OK » doivent être considérées comme des indications de réussite équivalentes, même si les phrases de raison suggèrent le contraire.

en-têtes

Zéro en-tête, un ou plusieurs en-têtes, chacun étant un nom, suivi de deux points (:), d'un espace facultatif, d'une valeur et d'un CRLF. Les en-têtes se terminent par une ligne vide (CRLF), marquant la fin de la liste des en-têtes et le début du corps de l'entité. Certaines versions de HTTP, comme HTTP/1.1, requièrent la présence de certains en-têtes pour que le message de requête ou de réponse soit valide. Les différents en-têtes HTTP sont abordés plus loin dans ce chapitre.

entité-corps

Le corps de l'entité contient un bloc de données arbitraires. Tous les messages ne contiennent pas de corps d'entité ; il arrive donc qu'un message se termine par un CRLF simple. Nous détaillons les entités au chapitre 15.

La figure 3-5 illustre des messages de demande et de réponse hypothétiques.

(a) Message de demande (b) Message de réponse

HTTP/1.0 200 OK

Salut ! Je suis un message !

OBTENIR /test/hi-there.txt HTTP/1.1

Ligne de départ

En-têtes

Corps

Figure 3-5. Exemples de messages de requête et de réponse

Notez qu'un ensemble d'en-têtes HTTP doit toujours se terminer par une ligne vide (CRLF nu), même en l'absence d'en-têtes et de corps d'entité. Cependant, historiquement, de nombreux clients et serveurs omettaient (par erreur) le CRLF final en l'absence de corps d'entité. Pour interagir avec ces implémentations courantes mais non conformes, les clients et les serveurs doivent accepter les messages se terminant sans le CRLF final.

Lignes de départ

Tous les messages HTTP commencent par une ligne de départ. La ligne de départ d'une requête indique la procédure à suivre. La ligne de départ d'une réponse indique ce qui s'est passé.

Ligne de demande

Les messages de requête demandent aux serveurs d'effectuer une action sur une ressource. La ligne de départ d'un message de requête, ou ligne de requête, contient une méthode décrivant l'opération à effectuer par le serveur et une URL de requête décrivant la ressource sur laquelle exécuter la méthode. La ligne de requête inclut également une version HTTP qui indique au serveur le dialecte HTTP utilisé par le client.

Tous ces champs sont séparés par des espaces. Dans la figure 3-5a, la méthode de requête est GET, l'URL de requête est /test/hi-there.txt et la version est HTTP/1.1. Avant HTTP/1.0, les lignes de requête n'étaient pas obligées de contenir une version HTTP.

Ligne de réponse

Les messages de réponse transmettent au client les informations d'état et les données résultant d'une opération. La ligne de début d'un message de réponse, ou ligne de réponse, contient la version HTTP utilisée, un code d'état numérique et une phrase explicative décrivant l'état de l'opération.

Tous ces champs sont séparés par des espaces. Dans la figure 3-5b, la version HTTP est HTTP/1.0, le code d'état est 200 (indiquant la réussite) et la phrase de raison est OK, ce qui signifie que le document a été renvoyé avec succès. Avant HTTP/1.0, les réponses n'étaient pas obligatoirement accompagnées d'une ligne de réponse.

Méthodes

La méthode commence la ligne de départ des requêtes, indiquant au serveur la marche à suivre. Par exemple, dans la ligne « GET /specials/saw-blade.gif HTTP/1.0 », la méthode est GET.

Les spécifications HTTP définissent un ensemble de méthodes de requête courantes. Par exemple, la méthode GET récupère un document auprès d'un serveur, la méthode POST envoie des données à un serveur pour traitement, et la méthode OPTIONS détermine les capacités générales d'un serveur web ou celles d'un serveur web pour une ressource spécifique.

Le tableau 3-1 décrit sept de ces méthodes. Notez que certaines méthodes ont un corps dans le message de requête, tandis que d'autres ont des requêtes sans corps.

Tableau 3-1. Méthodes HTTP courantes

Description de la méthode Corps du message ?

OBTENIR Obtenir un document du serveur. Non

HEAD Récupérez uniquement les en-têtes d'un document à partir du serveur. Non

POST : Envoi des données au serveur pour traitement. Oui

PUT Stocker le corps de la requête sur le serveur. Oui

TRACE Trace le message via des serveurs proxy jusqu'au serveur. Non

OPTIONS Déterminer quelles méthodes peuvent fonctionner sur un serveur. Non

SUPPRIMER Supprimer un document du serveur. Non

Tous les serveurs n'implémentent pas les sept méthodes du tableau 3-1. De plus, HTTP étant conçu pour être facilement extensible, d'autres serveurs peuvent implémenter leurs propres méthodes de requête en plus de celles-ci. Ces méthodes supplémentaires sont appelées méthodes d'extension, car elles étendent la spécification HTTP.

Codes d'état

Tandis que les méthodes indiquent au serveur ce qu'il doit faire, les codes d'état indiquent au client ce qui s'est passé. Ces codes figurent dans les premières lignes des réponses. Par exemple, à la ligne « HTTP/1.0 200 OK », le code d'état est 200.

Lorsque des clients envoient des messages de requête à un serveur HTTP, plusieurs événements peuvent se produire. Avec un peu de chance, la requête aboutira. Ce n'est pas toujours le cas. Le serveur peut vous indiquer que la ressource demandée est introuvable, que vous n'avez pas l'autorisation d'y accéder, ou qu'elle a été déplacée.

Des codes d'état sont renvoyés dans la première ligne de chaque message de réponse. Ils renvoient un état numérique et un état lisible par l'utilisateur. Le code numérique facilite le traitement des erreurs par les programmes, tandis que la phrase de raison est facilement compréhensible par l'utilisateur.

Les différents codes d'état sont regroupés en classes selon leurs codes numériques à trois chiffres. Les codes compris entre 200 et 299 indiquent une réussite. Les codes compris entre 300 et 399 indiquent que la ressource a été déplacée. Les codes compris entre 400 et 499 indiquent que le client a effectué une erreur dans la requête. Les codes compris entre 500 et 599 indiquent qu'un problème s'est produit sur le serveur.

Les classes de codes d’état sont présentées dans le tableau 3-2.

Tableau 3-2. Classes de codes d'état

Gamme globale Gamme définie Catégorie

100-199 100-101 Informatif

200-299 200-206 Réussi

300-399 300-305 Redirection

400-499 400-415 Erreur client

500-599 500-505 Erreur du serveur

Les versions actuelles de HTTP ne définissent que quelques codes pour chaque catégorie d'état. À mesure que le protocole évolue, davantage de codes d'état seront officiellement définis dans la spécification HTTP. Si vous recevez un code d'état que vous ne reconnaissez pas, il est probable qu'il ait été défini comme une extension du protocole actuel. Vous devez le traiter comme un membre général de la classe dont il fait partie.

Par exemple, si vous recevez le code d'état 515 (qui est en dehors de la plage définie pour les codes 5XX répertoriés dans le tableau 3-2), vous devez traiter la réponse comme indiquant une erreur de serveur, qui est la classe générale des messages 5XX.

Le tableau 3-3 répertorie certains des codes d'état HTTP les plus courants. Nous détaillerons tous les codes d'état HTTP actuels plus loin dans ce chapitre.

Tableau 3-3. Codes d'état courants

Code d'état Phrase de raison Signification

200 OK Succès ! Toutes les données demandées se trouvent dans le corps de la réponse.

401 Non autorisé Vous devez saisir un nom d'utilisateur et un mot de passe.

404 Non trouvé Le serveur ne trouve pas de ressource pour l'URL demandée.

Phrases de raison

La phrase de raison est le dernier élément de la ligne de début de la réponse. Elle fournit une explication textuelle du code d'état. Par exemple, à la ligne « HTTP/1.0 200 OK », la phrase de raison est OK.

Les phrases de raisonnement sont associées individuellement aux codes d'état. Elles fournissent une version lisible du code d'état que les développeurs d'applications peuvent transmettre à leurs utilisateurs pour indiquer le déroulement de la requête.

La spécification HTTP ne fournit aucune règle absolue quant à la forme que doivent prendre les phrases de raison. Plus loin dans ce chapitre, nous listons les codes d'état et quelques suggestions de phrases de raison.

Numéros de version

Les numéros de version apparaissent dans les lignes de démarrage des messages de demande et de réponse au format HTTP/xy. Ils fournissent aux applications HTTP un moyen de se dire mutuellement à quelle version du protocole elles se conforment.

Les numéros de version sont destinés à fournir aux applications utilisant HTTP des informations sur leurs capacités respectives et le format du message. Une application HTTP version 1.2 communiquant avec une application HTTP version 1.1 doit savoir qu'elle ne doit pas utiliser les nouvelles fonctionnalités de la version 1.2, car elles ne sont probablement pas implémentées par l'application utilisant l'ancienne version du protocole.

Le numéro de version indique la version la plus récente de HTTP prise en charge par une application. Cela peut parfois prêter à confusion entre les applications*, car les applications HTTP/1.0 interprètent une réponse contenant HTTP/1.1 comme une réponse 1.1, alors qu'il s'agit simplement du niveau de protocole utilisé par l'application qui répond.

Notez que les numéros de version ne sont pas traités comme des nombres fractionnaires. Chaque numéro de version (par exemple, le « 1 » et le « 0 » dans HTTP/1.0) est traité comme un numéro distinct. Ainsi, lors de la comparaison de versions HTTP, chaque numéro doit être comparé séparément.

* Voir http://httpd.apache.org/docs-2.0/misc/known_client_problems.html pour plus d'informations sur les cas dans lesquels Apache a rencontré ce problème avec des clients.

afin de déterminer quelle est la version la plus récente. Par exemple, HTTP/2.22 est une version plus récente que HTTP/2.3, car 22 est un nombre supérieur à 3.

En-têtes

La section précédente s'est concentrée sur la première ligne des messages de requête et de réponse (méthodes, codes d'état, phrases de raison et numéros de version). Après la ligne de départ se trouve une liste de zéro, un ou plusieurs champs d'en-tête HTTP (voir figure 3-5).

Les champs d'en-tête HTTP ajoutent des informations supplémentaires aux messages de requête et de réponse. Il s'agit simplement de listes de paires nom/valeur. Par exemple, la ligne d'en-tête suivante attribue la valeur 19 au champ d'en-tête Content-Length :

Longueur du contenu : 19

Classifications d'en-tête

La spécification HTTP définit plusieurs champs d'en-tête. Les applications sont également libres d'inventer leurs propres en-têtes. Les en-têtes HTTP sont classés comme suit :

En-têtes généraux

Peut apparaître dans les messages de demande et de réponse

En-têtes de requête

Fournir plus d'informations sur la demande

En-têtes de réponse

Fournir plus d'informations sur la réponse

En-têtes d'entité

Décrivez la taille du corps et son contenu, ou la ressource elle-même

En-têtes d'extension

Nouveaux en-têtes qui ne sont pas définis dans la spécification

Chaque en-tête HTTP possède une syntaxe simple : un nom, suivi de deux points (:), d'un espace facultatif, de la valeur du champ et d'un CRLF. Le tableau 3-4 présente quelques exemples d'en-têtes courants.

Tableau 3-4. Exemples d'en-têtes courants

Exemple d'en-tête Description

Date : mar. 3 oct. 1997 02:16:03 GMT La date à laquelle le serveur a généré la réponse

Longueur du contenu : 15 040 Le corps de l'entité contient 15 040 octets de données

Type de contenu : image/gif Le corps de l'entité est une image GIF

Accepter : image/gif, image/jpeg, texte/html Le client accepte les images GIF et JPEG et HTML

Lignes de continuation d'en-tête

Les longues lignes d'en-tête peuvent être rendues plus lisibles en les divisant en plusieurs lignes, précédant chaque ligne supplémentaire par au moins un espace ou un caractère de tabulation.

Par exemple:

HTTP/1.0 200 OK

Type de contenu : image/gif

Contenu-Longueur : 8572

Serveur : Serveur de test

Version 1.0

Dans cet exemple, le message de réponse contient un en-tête « Serveur » dont la valeur est divisée en lignes de continuation. La valeur complète de l'en-tête est « Test Server Version 1.0 ».

Nous décrirons brièvement tous les en-têtes HTTP plus loin dans ce chapitre. Un résumé de référence plus détaillé de tous les en-têtes est également fourni dans l'annexe C.

Organismes d'entité

La troisième partie d'un message HTTP est le corps d'entité facultatif. Les corps d'entité constituent la charge utile des messages HTTP. Ce sont les éléments que HTTP a été conçu pour transporter.

Les messages HTTP peuvent transporter de nombreux types de données numériques : images, vidéos, documents HTML, applications logicielles, transactions par carte de crédit, courrier électronique, etc.

Messages de la version 0.9

HTTP version 0.9 était une version précoce du protocole HTTP. Elle a servi de point de départ aux messages de requête et de réponse actuels du protocole HTTP, mais avec un protocole beaucoup plus simple (voir figure 3-6).

Figure 3-6. Transaction HTTP/0.9

Les messages HTTP/0.9 étaient également composés de requêtes et de réponses, mais la requête contenait uniquement la méthode et l'URL de la requête, et la réponse ne contenait que l'entité. Aucune information de version (il s'agissait de la première et unique version à ce moment-là), aucun code d'état ni aucune phrase de raison, et aucun en-tête n'étaient inclus.

Cependant, cette simplicité ne permettait pas une grande flexibilité ni l'implémentation de la plupart des fonctionnalités et applications HTTP décrites dans ce livre. Nous la décrivons brièvement ici, car des clients, des serveurs et d'autres applications l'utilisent encore, et les développeurs d'applications doivent être conscients de ses limites.

Méthodes

Examinons plus en détail certaines des méthodes HTTP de base, répertoriées précédemment dans le tableau 3-1. Notez que toutes les méthodes ne sont pas implémentées par tous les serveurs. Pour être conforme à la version 1.1 de HTTP, un serveur doit implémenter uniquement les méthodes GET et HEAD pour ses ressources.

Même lorsque les serveurs implémentent toutes ces méthodes, leur utilisation est probablement restreinte. Par exemple, les serveurs prenant en charge les méthodes DELETE ou PUT (décrites plus loin dans cette section) ne souhaitent pas que n'importe qui puisse supprimer ou stocker des ressources. Ces restrictions sont généralement définies dans la configuration du serveur et varient donc d'un site à l'autre et d'un serveur à l'autre.

Méthodes sûres

HTTP définit un ensemble de méthodes dites sûres. Les méthodes GET et HEAD sont dites sûres, ce qui signifie qu'aucune action ne doit être effectuée suite à une requête HTTP utilisant l'une ou l'autre de ces méthodes.

Par « aucune action », nous entendons qu'aucune action ne se produira sur le serveur suite à la requête HTTP. Par exemple, lorsque vous effectuez un achat en ligne chez Joe's Hardware, vous cliquez sur le bouton « Soumettre l'achat ». Ce clic envoie une requête POST (voir plus loin) contenant vos informations de carte bancaire, et une action est effectuée sur le serveur en votre nom. Dans ce cas, votre carte bancaire est débitée de votre achat.

Il n'y a aucune garantie qu'une méthode sûre n'entraîne pas l'exécution d'une action (en pratique, c'est aux développeurs web de le faire). Les méthodes sûres permettent aux développeurs d'applications HTTP d'avertir les utilisateurs lorsqu'une méthode non sûre est utilisée et peut entraîner l'exécution d'une action. Dans notre exemple, Joe's Hardware, votre navigateur web peut afficher un message

d'avertissement vous informant que vous effectuez une requête avec une méthode non sûre et qu'un incident risque de se produire sur le serveur (par exemple, le débit de votre carte de crédit).

OBTENIR

GET est la méthode la plus courante. Elle est généralement utilisée pour demander à un serveur d'envoyer une ressource. HTTP/1.1 exige que les serveurs implémentent cette méthode. La figure 3-7 montre un exemple de requête HTTP effectuée par un client avec la méthode GET.

Figure 3-7. Exemple GET

TÊTE

La méthode HEAD se comporte exactement comme la méthode GET, mais le serveur ne renvoie que les en-têtes de la réponse. Aucun corps d'entité n'est renvoyé. Cela permet au client d'inspecter les en-têtes d'une ressource sans avoir à l'obtenir. Avec HEAD, vous pouvez :

• Se renseigner sur une ressource (par exemple, déterminer son type) sans l'obtenir.

• Vérifiez si un objet existe en regardant le code d'état de la réponse.

• Testez si la ressource a été modifiée, en regardant les en-têtes.

Les développeurs de serveurs doivent s'assurer que les en-têtes renvoyés correspondent exactement à ceux d'une requête GET. La méthode HEAD est également requise pour la conformité HTTP/1.1. La figure 3-8 illustre la méthode HEAD en action.

Figure 3-8. Exemple de TÊTE

METTRE

La méthode PUT écrit les documents sur un serveur, à l'inverse de la méthode GET qui lit les documents depuis un serveur. Certains systèmes de publication permettent de créer des pages web et de les installer directement sur un serveur web grâce à PUT (voir figure 3-9).

Figure 3-9. Exemple PUT

La sémantique de la méthode PUT permet au serveur de prendre le corps de la requête et de l'utiliser pour créer un nouveau document nommé par l'URL demandée ou, si cela

L'URL existe déjà, utilisez le corps pour la remplacer.

Comme PUT permet de modifier le contenu, de nombreux serveurs web exigent une connexion avec un mot de passe avant de pouvoir effectuer une opération PUT. Pour en savoir plus sur l'authentification par mot de passe, consultez le chapitre 12.

POSTE

La méthode POST a été conçue pour envoyer des données d'entrée au serveur.* En pratique, elle est souvent utilisée pour gérer les formulaires HTML. Les données d'un formulaire rempli sont généralement envoyées au serveur, qui les achemine ensuite vers l'emplacement souhaité (par exemple, vers une passerelle serveur, qui les traite ensuite). La figure 3-10 illustre un client effectuant une requête HTTP (envoyant des données de formulaire à un serveur) avec la méthode POST.

TRACER

Lorsqu'un client effectue une requête, celle-ci peut être amenée à transiter par des pare-feu, des proxys, des passerelles ou d'autres applications. Chacun de ces éléments peut modifier la requête HTTP d'origine. La méthode TRACE permet aux clients de visualiser l'état de leur requête une fois celle-ci parvenue au serveur.

Une requête TRACE déclenche un diagnostic de « bouclage » sur le serveur de destination. Le serveur de la dernière étape du trajet renvoie une réponse TRACE, avec le message vierge.

* POST est utilisé pour envoyer des données à un serveur. PUT est utilisé pour déposer des données dans une ressource du serveur (par exemple, un fichier).

Figure 3-10. Exemple POST

Le client peut alors voir si son message d'origine a été modifié tout au long de la chaîne requête/réponse des applications HTTP intervenantes (voir figure 3-11).

La méthode TRACE est principalement utilisée à des fins de diagnostic, c'est-à-dire pour vérifier que les requêtes suivent la chaîne requête/réponse comme prévu. C'est également un bon outil pour observer l'impact des proxys et autres applications sur vos requêtes.

Aussi efficace que soit TRACE pour le diagnostic, il présente l'inconvénient de supposer que les applications intermédiaires traiteront de la même manière différents types de requêtes (différentes méthodes : GET, HEAD, POST, etc.). De nombreuses applications HTTP fonctionnent différemment selon la méthode. Par exemple, un proxy peut transmettre une requête POST directement au serveur, mais tenter d'envoyer une requête GET à une autre application HTTP (comme un cache web). TRACE ne fournit pas de mécanisme

permettant de distinguer les méthodes. Généralement, ce sont les applications intermédiaires qui décident comment traiter une requête TRACE.

demande.

Figure 3-11. Exemple TRACE

Aucun corps d'entité ne peut être envoyé avec une requête TRACE. Le corps d'entité de la réponse TRACE contient, textuellement, la requête reçue par le serveur répondant.

OPTIONS

La méthode OPTIONS demande au serveur de nous informer sur les différentes fonctionnalités prises en charge par le serveur web. Vous pouvez lui demander quelles méthodes il prend en charge, en général ou pour des ressources spécifiques. (Certains serveurs peuvent prendre en charge certaines opérations uniquement sur certains types d'objets).

Cela permet aux applications clientes de déterminer la meilleure façon d'accéder à diverses ressources sans avoir à y accéder. La figure 3-12 illustre un scénario de requête utilisant la méthode OPTIONS.

Figure 3-12. Exemple d'OPTIONS

SUPPRIMER

La méthode DELETE fait exactement ce que l'on pourrait croire : elle demande au serveur de supprimer les ressources spécifiées par l'URL de la requête. Cependant, l'application cliente n'a aucune garantie que la suppression soit effectuée. En effet, la spécification HTTP permet au serveur d'ignorer la requête sans en informer le client. La figure 3-13 présente un exemple de la méthode DELETE.

Figure 3-13. Exemple de SUPPRESSION

Méthodes d'extension

HTTP a été conçu pour être extensible sur le terrain, afin que les nouvelles fonctionnalités n'entraînent pas de pannes d'anciens logiciels. Les méthodes d'extension ne sont pas définies dans la spécification HTTP/1.1. Elles permettent aux développeurs d'étendre les capacités des services HTTP implémentés par leurs serveurs sur les ressources gérées par ces derniers. Le tableau 3-5 présente quelques exemples courants de méthodes d'extension. Ces méthodes font toutes

partie de l'extension HTTP WebDAV (voir chapitre 19), qui permet la publication de contenu web sur des serveurs web via HTTP.

Tableau 3-5. Exemples de méthodes d'extension de publication Web

Description de la méthode

VERROUILLER Permet à un utilisateur de « verrouiller » une ressource. Par exemple, vous pouvez verrouiller une ressource pendant que vous la modifiez pour empêcher d'autres personnes de la modifier en même temps.

MKCOL permet à un utilisateur de créer une ressource

COPY Facilite la copie des ressources sur un serveur

MOVE Déplace une ressource sur un serveur

Il est important de noter que toutes les méthodes d'extension ne sont pas définies dans une spécification formelle. Si vous définissez une méthode d'extension, elle risque de ne pas être comprise par la plupart des applications HTTP. De même, il est possible que vos applications HTTP rencontrent des méthodes d'extension utilisées par d'autres applications qu'elles ne comprennent pas.

Dans ces cas-là, il est préférable de faire preuve de tolérance envers les méthodes d'extension. Les proxys doivent tenter de relayer les messages utilisant des méthodes inconnues vers les serveurs en aval s'ils en sont capables sans perturber le comportement de bout en bout. Dans le cas contraire, ils doivent renvoyer un code d'état 501 « Non implémenté ». La gestion des méthodes d'extension (et des extensions HTTP en général) repose sur la règle classique : « Soyez prudent dans ce que vous envoyez, soyez généreux dans ce que vous acceptez. »

Codes d'état

Les codes d'état HTTP sont classés en cinq grandes catégories, comme indiqué précédemment dans

Tableau 3-2. Cette section résume les codes d'état HTTP pour chacune des cinq classes.

Les codes d'état permettent aux clients de comprendre facilement les résultats de leurs transactions. Cette section propose également des exemples de phrases de raisonnement, sans toutefois fournir de précisions sur leur texte exact. Nous incluons les phrases de raisonnement recommandées par la spécification HTTP/1.1.

100–199 : Codes d'état informatifs

HTTP/1.1 a introduit les codes d'état informatifs dans le protocole. Relativement nouveaux, ils suscitent une certaine controverse quant à leur complexité et à leur valeur perçue. Le tableau 3-6 répertorie les codes d'état informatifs définis.

Tableau 3-6. Codes d'état informatifs et phrases de raison

Code d'état Phrase de raison Signification

100 Continuer : indique qu'une partie initiale de la requête a été reçue et que le client doit continuer. Après l'envoi, le serveur doit répondre après avoir reçu la requête. Pour plus d'informations, consultez l'en-tête « Expect » de l'annexe C.

101 Protocoles de commutation Indique que le serveur change de protocole, comme spécifié par le client, vers un protocole répertorié dans l'en-tête de mise à niveau.

Le code d'état 100 Continue, en particulier, est un peu déroutant. Il vise à optimiser le cas où une application cliente HTTP doit envoyer un corps d'entité à un serveur, mais souhaite vérifier que le serveur l'accepte avant de l'envoyer. Nous l'abordons ici plus en détail (son interaction avec les clients, les serveurs et les proxys), car il peut perturber les programmeurs HTTP.

Clients et 100 Continuer

Si un client envoie une entité à un serveur et accepte d'attendre une réponse 100 Continue avant de l'envoyer, il doit envoyer un en-tête de requête Expect (voir Annexe C) avec la valeur 100-continue. Si le client n'envoie pas d'entité, il ne doit pas envoyer d'en-tête Expect 100-continue, car cela ne ferait qu'induire le serveur en erreur.

100-continue est, à bien des égards, une optimisation. Une application cliente ne devrait utiliser 100-continue que pour éviter d'envoyer au serveur une entité volumineuse que celui-ci ne pourra ni gérer ni utiliser.

En raison de la confusion initiale autour du statut 100 Continue (et compte tenu de certaines des implémentations les plus anciennes), les clients qui envoient un en-tête Expect pour 100continue ne doivent pas attendre indéfiniment que le serveur envoie une réponse 100 Continue.

Après un certain délai, le client doit simplement envoyer l'entité.

En pratique, les développeurs clients doivent également se préparer à gérer des réponses 100 Continue inattendues (c'est ennuyeux, mais vrai). Certaines applications HTTP erronées envoient ce code de manière inappropriée.

Serveurs et 100 Continuer

Si un serveur reçoit une requête avec l'en-tête Expect et la valeur 100-continue, il doit répondre soit par la réponse 100 Continue, soit par un code d'erreur (voir Tableau 3-9). Les serveurs ne doivent jamais envoyer un code d'état 100 Continue aux clients qui n'envoient pas

l'attente 100-continue. Cependant, comme indiqué précédemment, certains serveurs errants le font.

Si, pour une raison quelconque, le serveur reçoit une partie (ou la totalité) de l'entité avant d'avoir pu envoyer une réponse « 100 Continue », il n'a pas besoin d'envoyer ce code d'état, car le client a déjà décidé de continuer. Cependant, une fois la lecture de la requête terminée, le serveur doit encore envoyer un code d'état final (il peut simplement ignorer l'état « 100 Continue »).

Enfin, si un serveur reçoit une requête avec une attente de 100-continue et qu'il décide de terminer la requête avant d'avoir lu le corps de l'entité (par exemple, parce qu'une erreur s'est produite), il ne doit pas simplement envoyer une réponse et fermer la connexion, car cela peut empêcher le client de recevoir la réponse (voir « Erreurs de fermeture et de réinitialisation TCP » au chapitre 4).

Procurations et 100 Continuer

Un proxy recevant d'un client une requête contenant l'attente 100-continue doit effectuer plusieurs opérations. Si le proxy sait que le serveur de saut suivant (abordé au chapitre 6) est compatible HTTP/1.1 ou ignore sa version, il doit transmettre la requête avec l'en-tête Expect. S'il sait que le serveur de saut suivant est compatible avec une version de HTTP antérieure à 1.1, il doit renvoyer l'erreur 417 Expectation Failed.

Si un proxy décide d'inclure un en-tête Expect et une valeur 100-continue dans sa demande au nom d'un client conforme à HTTP/1.0 ou une version antérieure, il ne doit pas transmettre la réponse 100 Continue (s'il en reçoit une du serveur) au client, car le client ne saura pas quoi en faire.

Il peut être avantageux pour les proxys de maintenir un certain état sur les serveurs de saut suivant et les versions de HTTP qu'ils prennent en charge (au moins pour les serveurs qui ont reçu des requêtes récentes), afin qu'ils puissent mieux gérer les requêtes reçues avec une attente de 100 continues.

200–299 : Codes d'état de réussite

Les requêtes des clients aboutissent généralement. Les serveurs disposent d'un ensemble de codes d'état indiquant la réussite, correspondant à différents types de requêtes. Le tableau 3-7 répertorie les codes d'état de réussite définis.

Tableau 3-7. Codes de réussite et phrases de raison

Code d'état Phrase de raison Signification

200 OK La demande est correcte, le corps de l'entité contient la ressource demandée.

201 Créé : pour les requêtes créant des objets serveur (par exemple, PUT). Le corps de la réponse doit contenir les différentes URL de référencement de la ressource créée, l'en-tête Location contenant la référence la plus spécifique. Voir le tableau 3-21 pour plus d'informations sur l'en-tête Location.

Le serveur doit avoir créé l'objet avant d'envoyer ce code d'état.

202 Acceptée : la requête a été acceptée, mais le serveur n'a encore effectué aucune action. Rien ne garantit que le serveur traitera la requête ; cela signifie simplement que la requête semblait valide au moment de son acceptation.

Le serveur doit inclure un corps d'entité avec une description indiquant l'état de la demande et éventuellement une estimation du moment où elle sera terminée (ou un pointeur vers l'endroit où ces informations peuvent être obtenues).

203 Non autorisé

Informations : les informations contenues dans les en-têtes d'entité (voir « En-têtes d'entité » pour plus d'informations sur les en-têtes d'entité) ne proviennent pas du serveur d'origine, mais d'une copie de la ressource. Cela peut se produire si un intermédiaire possède une copie d'une ressource, mais ne peut pas valider les méta-informations (en-têtes) qu'il envoie à son sujet.

Ce code de réponse n'est pas obligatoire ; il s'agit d'une option pour les applications dont la réponse serait un statut 200 si les en-têtes d'entité provenaient du serveur d'origine.

204 Aucun contenu. Le message de réponse contient des en-têtes et une ligne d'état, mais pas de corps d'entité. Principalement utilisé pour mettre à jour les navigateurs sans les déplacer vers un nouveau document (par exemple, pour actualiser une page de formulaire).

205 Réinitialiser le contenu Un autre code principalement destiné aux navigateurs. Indique au navigateur d'effacer tous les éléments de formulaire HTML de la page actuelle.

206 Contenu partiel : une requête partielle ou étendue a réussi. Nous verrons plus tard que les clients peuvent demander une partie ou une étendue d'un document en utilisant des en-têtes spéciaux. Ce code d'état indique que la requête a réussi. Voir « Requêtes étendues » au chapitre 15 pour plus d'informations sur l'en-tête « Étendue ».

Une réponse 206 doit inclure un en-tête Content-Range, Date et ETag ou ContentLocation.

300–399 : Codes d'état de redirection

Les codes d'état de redirection indiquent aux clients d'utiliser d'autres emplacements pour les ressources qui les intéressent ou fournissent

une réponse alternative au lieu du contenu. Si une ressource a été déplacée, un code d'état de redirection et un en-tête d'emplacement facultatif peuvent être envoyés pour informer le client que la ressource a été déplacée et où elle se trouve désormais (voir la figure 3-14). Cela permet aux navigateurs d'accéder au nouvel emplacement de manière transparente, sans gêner les utilisateurs.

Figure 3-14. Demande redirigée vers un nouvel emplacement

Certains codes d'état de redirection peuvent être utilisés pour valider la copie locale d'une ressource d'une application auprès du serveur d'origine. Par exemple, une application HTTP peut vérifier si la copie locale de sa ressource est toujours à jour ou si la ressource a été modifiée sur le serveur d'origine. La figure 3-15 illustre ce processus. Le client envoie un en-tête spécial If-Modified-Since indiquant de ne récupérer le document que s'il a été modifié depuis octobre 1997. Le document n'ayant pas été modifié depuis cette date, le serveur renvoie un code d'état 304 au lieu du contenu.

Figure 3-15. Requête redirigée vers une copie locale

En général, il est recommandé que les réponses aux requêtes non HEAD incluant un code d'état de redirection incluent une entité avec une description et des liens vers les URL redirigées (voir le premier message de réponse de la figure 3-14). Le tableau 3-8 répertorie les codes d'état de redirection définis.

Tableau 3-8. Codes d'état de redirection et phrases de raison

Code d'état Phrase de raison Signification

300 Choix multiples renvoyés lorsqu'un client demande une URL qui renvoie en réalité à plusieurs ressources, comme un serveur hébergeant les versions anglaise et française d'un document HTML. Ce code est renvoyé avec une liste d'options ; l'utilisateur peut ensuite sélectionner celle qu'il souhaite. Consultez le chapitre 17 pour en savoir plus sur la négociation entre clients et versions multiples. Le serveur peut inclure l'URL souhaitée dans l'en-tête « Emplacement ».

301 Déplacé définitivement. Utilisé lorsque l'URL demandée a été déplacée. La réponse doit contenir, dans l'en-tête Emplacement, l'URL où se trouve désormais la ressource.

Code d'état 302 Trouvé : similaire au code d'état 301. Cependant, le client doit utiliser l'URL indiquée dans l'en-tête Emplacement pour localiser temporairement la ressource. Les requêtes ultérieures devront utiliser l'ancienne URL.

Tableau 3-8. Codes d'état de redirection et phrases de raison (suite)

Code d'état	Phrase de raison	Signification
303	Voir Autre	utilisé pour indiquer au client que la ressource doit être récupérée via une URL différente. Cette nouvelle URL se trouve dans l'en-tête « Location » du message de réponse. Son objectif principal est de permettre aux réponses aux requêtes POST de diriger le client vers une ressource.
304	Non modifié	les clients peuvent conditionner leurs requêtes grâce aux en-têtes de requête inclus. Consultez le tableau 3-15 pour plus d'informations sur les en-têtes conditionnels. Si un client effectue une requête conditionnelle, telle qu'une requête GET si la ressource n'a pas été modifiée récemment, ce code indique que la ressource n'a pas été modifiée. Les réponses avec ce code d'état ne doivent pas contenir de corps d'entité.
305	Utiliser un proxy	indique que la ressource doit être accessible via un proxy ; l'emplacement du proxy est indiqué dans l'en-tête « Location ». Il est important que les clients interprètent cette réponse par rapport à une ressource spécifique et ne supposent pas que ce proxy doit être utilisé pour toutes les requêtes, ni même pour toutes les requêtes adressées au serveur hébergeant la ressource demandée. Cela pourrait entraîner un dysfonctionnement si le proxy interférait par erreur avec une requête, et constituer une faille de sécurité.
306	(inutilisé)	Non utilisé actuellement.
Redirection temporaire	307	comme le code d'état 301, le client doit utiliser l'URL indiquée dans l'en-tête Location pour localiser temporairement la ressource. Les requêtes futures devront utiliser l'ancienne URL.

D'après le tableau 3-8, vous avez peut-être remarqué un léger chevauchement entre les codes d'état 302, 303 et 307. Leur utilisation présente quelques nuances, principalement dues aux différences de traitement entre les applications HTTP/1.0 et HTTP/1.1.

Lorsqu'un client HTTP/1.0 effectue une requête POST et reçoit un code d'état de redirection 302 en réponse, il suivra l'URL de redirection dans l'en-tête Location avec une requête GET vers cette URL (au lieu d'effectuer une requête POST, comme il l'a fait dans la requête d'origine).

Les serveurs HTTP/1.0 s'attendent à ce que les clients HTTP/1.0 fassent cela : lorsqu'un serveur HTTP/1.0 envoie un code d'état 302 après avoir reçu une requête POST d'un client HTTP/1.0, le serveur s'attend à ce que ce client suive la redirection avec une requête GET vers le client redirigé.

URL.

La confusion vient de HTTP/1.1. La spécification HTTP/1.1 utilise le code d'état 303 pour obtenir le même comportement (les serveurs envoient le code d'état 303 pour rediriger la requête POST d'un client vers une requête GET).

Pour éviter toute confusion, la spécification HTTP/1.1 indique d'utiliser le code d'état 307 à la place du code d'état 302 pour les redirections temporaires vers les clients HTTP/1.1.

Les serveurs peuvent ensuite enregistrer le code d'état 302 pour l'utiliser avec les clients HTTP/1.0.

Tout cela revient à dire que les serveurs doivent vérifier la version HTTP d'un client pour sélectionner correctement le code d'état de redirection à envoyer dans une réponse de redirection.

400–499 : Codes d'état d'erreur du client

Parfois, un client envoie quelque chose qu'un serveur ne peut tout simplement pas gérer, comme un message de demande mal formé ou, le plus souvent, une demande d'URL qui n'existe pas.

Nous avons tous vu le tristement célèbre code d'erreur 404 Not Found lors de la navigation : il s'agit simplement du serveur qui nous indique que nous avons demandé une ressource dont il ne sait rien.

La plupart des erreurs client sont traitées par votre navigateur, sans que vous en soyez affecté. Certaines, comme 404, peuvent néanmoins passer inaperçues. Le tableau 3-9 présente les différents codes d'état d'erreur client.

Tableau 3-9. Codes d'état d'erreur du client et phrases de raison

Code d'état Phrase de raison Signification

400 Bad Request Utilisé pour indiquer au client qu'il a envoyé une demande mal formée.

401 Non autorisé : renvoyé avec les en-têtes appropriés demandant au client de s'authentifier avant d'accéder à la ressource. Voir le chapitre 12 pour plus d'informations sur l'authentification.

402 Paiement requis Actuellement, ce code de statut n'est pas utilisé, mais il a été mis de côté pour une utilisation future.

403 Interdit : utilisé pour indiquer que la requête a été refusée par le serveur. Si le serveur souhaite indiquer la raison du refus, il peut inclure un corps d'entité décrivant la raison. Cependant, ce code est généralement utilisé lorsque le serveur ne souhaite pas révéler la raison du refus.

404 Introuvable : indique que le serveur ne trouve pas l'URL demandée. Une entité est souvent incluse pour que l'application cliente l'affiche à l'utilisateur.

405 Méthode non autorisée : utilisé lorsqu'une requête utilise une méthode non prise en charge par l'URL demandée. L'en-tête Allow doit être inclus dans la réponse pour indiquer au client les méthodes autorisées sur la ressource demandée. Consultez la section « En-têtes d'entité » pour plus d'informations sur l'en-tête Allow.

406 Non acceptable : les clients peuvent spécifier des paramètres concernant les types d'entités qu'ils acceptent. Ce code est utilisé lorsque le serveur ne dispose d'aucune ressource correspondant à l'URL.

est acceptable pour le client. Souvent, les serveurs incluent des en-têtes permettant au client de comprendre pourquoi la requête n'a pas pu être satisfaite. Pour plus d'informations, consultez la section « Négociation et transcodage de contenu » au chapitre 17.

407 Authentification proxy

Obligatoire Comme le code d'état 401, mais utilisé pour les serveurs proxy qui nécessitent une authentification pour une ressource.

408 Délai d'expiration de la requête : si un client met trop de temps à traiter sa requête, un serveur peut renvoyer ce code d'état et fermer la connexion. La durée de ce délai varie d'un serveur à l'autre, mais est généralement suffisante pour répondre à toute requête légitime.

409 Conflit : utilisé pour indiquer un conflit que la requête pourrait engendrer sur une ressource. Les serveurs peuvent envoyer ce code lorsqu'ils craignent qu'une requête puisse engendrer un conflit. La réponse doit contenir un corps décrivant le conflit.

410 Gone Similaire à 404, sauf que le serveur détenait auparavant la ressource. Utilisé principalement pour la maintenance des sites web, afin que l'administrateur du serveur puisse avertir les clients lorsqu'une ressource a été supprimée.

Tableau 3-9. Codes d'état d'erreur client et phrases de raison (suite)

Code d'état Phrase de raison Signification

411 Longueur requise : utilisé lorsque le serveur exige un en-tête Content-Length dans le message de requête. Voir « En-têtes de contenu » pour plus d'informations sur l'en-tête Content-Length.

412 Échec de la précondition : utilisé si un client effectue une requête conditionnelle et que l'une des conditions échoue. Les requêtes conditionnelles se produisent lorsqu'un client inclut un en-tête Expect. Voir l'annexe C pour plus d'informations sur l'en-tête Expect.

413 Entité de demande trop grande Utilisé lorsqu'un client envoie un corps d'entité plus grand que ce que le serveur peut ou veut traiter.

414 URI de demande trop long Utilisé lorsqu'un client envoie une demande avec une URL de demande plus grande que ce que le serveur peut ou veut traiter.

415 Type de média non pris en charge Utilisé lorsqu'un client envoie une entité d'un type de contenu que le serveur ne comprend pas ou ne prend pas en charge.

416 Plage demandée non

Satisfaisant Utilisé lorsque le message de demande demandait une plage d'une ressource donnée et que cette plage n'était pas valide ou ne pouvait pas être respectée.

417 Échec de l'attente : utilisé lorsque la requête contenait une attente dans l'en-tête Expect que le serveur n'a pas pu satisfaire. Voir l'annexe C pour plus d'informations sur l'en-tête Expect.

Un proxy ou une autre application intermédiaire peut envoyer ce code de réponse s'il dispose d'une preuve sans ambiguïté que le serveur d'origine générera une attente échouée pour la demande.

500–599 : Codes d'état d'erreur du serveur

Il arrive qu'un client envoie une requête valide, mais que le serveur lui-même présente une erreur. Il peut s'agir d'une limitation du serveur ou d'une erreur dans l'un des sous-composants du serveur, comme une ressource de passerelle.

Les proxys rencontrent souvent des problèmes lorsqu'ils tentent de communiquer avec les serveurs pour le compte d'un client. Ils génèrent des codes d'erreur serveur 5XX pour décrire le problème (le chapitre 6 détaille ce point). Le tableau 3-10 répertorie les codes d'erreur serveur définis.

Tableau 3-10. Codes d'état d'erreur du serveur et phrases de raison

Code d'état Phrase de raison Signification

500 Erreur interne du serveur Utilisé lorsque le serveur rencontre une erreur qui l'empêche de traiter la demande.

501 Non implémenté Utilisé lorsqu'un client fait une demande qui dépasse les capacités du serveur (par exemple, en utilisant une méthode de demande que le serveur ne prend pas en charge).

502 Bad Gateway Utilisé lorsqu'un serveur agissant comme proxy ou passerelle rencontre une fausse réponse du lien suivant dans la chaîne de réponse de la demande (par exemple, s'il ne parvient pas à se connecter à sa passerelle parente).

503 Service indisponible : indique que le serveur ne peut actuellement pas traiter la requête, mais qu'il le pourra ultérieurement. Si le serveur sait quand la ressource sera disponible, il peut inclure un en-tête

« Retry-After » dans la réponse. Voir « En-têtes de réponse » pour plus d'informations sur l'en-tête « Retry-After ».

Tableau 3-10. Codes d'état d'erreur du serveur et phrases explicatives (suite)

Code d'état Phrase de raison Signification

504 Délai d'expiration de la passerelle Similaire au code d'état 408, sauf que la réponse provient d'une passerelle ou d'un proxy qui a expiré en attendant une réponse à sa demande d'un autre serveur.

505 Version HTTP non

Utilisé lorsqu'un serveur reçoit une requête dans une version du protocole qu'il ne peut ou ne veut pas prendre en charge. Certaines applications serveur choisissent de ne pas prendre en charge les anciennes versions du protocole.

En-têtes

Les en-têtes et les méthodes fonctionnent ensemble pour déterminer les actions des clients et des serveurs. Cette section décrit brièvement les fonctions des en-têtes HTTP standard et de certains en-têtes non explicitement définis dans la spécification HTTP/1.1 (RFC 2616). L'annexe C résume tous ces en-têtes plus en détail.

Il existe des en-têtes spécifiques à chaque type de message, et des en-têtes plus généraux, fournissant des informations à la fois dans les messages de requête et de réponse. Les en-têtes se répartissent en cinq grandes catégories :

En-têtes généraux

Il s'agit d'en-têtes génériques utilisés par les clients et les serveurs. Ils ont une fonction générale et sont utiles aux clients, aux serveurs et aux autres applications pour s'échanger des informations. Par exemple, l'en-tête « Date » est un en-tête à usage général qui permet aux deux parties d'indiquer la date et l'heure de création du message :

Date : mar. 3 oct. 1974 02:16:00 GMT

En-têtes de requête

Comme leur nom l'indique, les en-têtes de requête sont spécifiques aux messages de requête. Ils fournissent des informations supplémentaires aux serveurs, comme le type de données que le client accepte de recevoir. Par exemple, l'en-tête Accept suivant indique au serveur que le client acceptera tout type de média correspondant à sa requête :

Accepter: */*

En-têtes de réponse

Les messages de réponse possèdent leurs propres en-têtes qui fournissent des informations au client (par exemple, le type de serveur auquel il communique). Par exemple, l'en-tête « Server » suivant indique au client qu'il communique avec un serveur Tiki-Hut version 1.0 :

Serveur : Tiki-Hut/1.0

En-têtes d'entité

Les en-têtes d'entité désignent les en-têtes qui traitent du corps de l'entité. Par exemple, ils peuvent indiquer le type des données qu'ils contiennent. Par exemple, l'en-tête Content-Type suivant indique à l'application que les données sont un document HTML utilisant le jeu de caractères iso-latin-1 :

Type de contenu : texte/html ; jeu de caractères = iso-latin-1

En-têtes d'extension

Les en-têtes d'extension sont des en-têtes non standard créés par les développeurs d'applications, mais non encore intégrés à la spécification HTTP officielle. Les programmes HTTP doivent tolérer et transmettre les en-têtes d'extension, même s'ils n'en connaissent pas la signification.

En-têtes généraux

Certains en-têtes fournissent des informations très basiques sur un message. Ces en-têtes sont appelés en-têtes généraux. Ils constituent des éléments clés, fournissant des informations utiles sur un message, quel que soit son type.

Par exemple, que vous construisiez un message de demande ou un message de réponse, la date et l'heure de création du message signifient la même chose, donc l'en-tête qui fournit ce type d'informations est général aux deux types de messages.

Le tableau 3-11 répertorie les en-têtes d'information générale. Tableau 3-11. En-têtes d'information générale

Description de l'en-tête

Connexion Permet aux clients et aux serveurs de spécifier des options concernant la connexion de demande/réponse

Datea fournit un horodatage indiquant quand le message a été créé

Version MIME Indique la version de MIME utilisée par l'expéditeur

Bande-annonce Répertorie l'ensemble des en-têtes qui se trouvent dans la bande-annonce d'un message codé avec l'encodage de transfert fragmentéb

Transfer-Encoding Indique au récepteur quel codage a été effectué sur le message afin qu'il soit transporté en toute sécurité

Mise à niveau Fournit une nouvelle version ou un nouveau protocole que l'expéditeur souhaite « mettre à niveau » en utilisant

Via Indique par quels intermédiaires (proxies, passerelles) le message est passé

a L'annexe C répertorie les formats de date acceptables pour l'en-tête Date. b Les codages de transfert fragmentés sont abordés plus en détail dans la section « Connexions fragmentées et persistantes » du chapitre 15.

En-têtes de mise en cache généraux

HTTP/1.0 a introduit les premiers en-têtes permettant aux applications HTTP de mettre en cache des copies locales d'objets au lieu de toujours les récupérer directement depuis le serveur d'origine. La dernière version de HTTP dispose d'un ensemble très complet de paramètres de cache. Le chapitre 7 aborde la mise en cache en détail. Le tableau 3-12 répertorie les en-têtes de mise en cache de base. Tableau 3-12. En-têtes de mise en cache généraux

Description de l'en-tête

Cache-Control Utilisé pour transmettre les instructions de mise en cache avec le message

Pragmaa Une autre façon de transmettre des instructions avec le message, bien que non spécifique à la mise en cache

Techniquement, un Pragma est un en-tête de requête. Son utilisation dans les réponses n'a jamais été spécifiée. En raison de son utilisation abusive et fréquente comme en-tête de réponse, de nombreux clients et proxys l'interprètent comme tel, mais sa sémantique précise n'est pas clairement définie. Quoi qu'il en soit, Pragma est déconseillé au profit de Cache-Control.

En-têtes de requête

Les en-têtes de requête sont des en-têtes qui n'ont de sens que dans un message de requête. Ils fournissent des informations sur l'expéditeur de la requête, son origine, ou les préférences et capacités du client. Les serveurs peuvent utiliser les informations que ces en-têtes leur fournissent sur le client pour tenter de lui fournir une meilleure réponse. Le tableau 3-13 répertorie les en-têtes d'information des requêtes. Tableau 3-13. En-têtes d'information des requêtes

Description de l'en-tête

Client-IPa Fournit l'adresse IP de la machine sur laquelle le client s'exécute

De Fournit l'adresse e-mail de l'utilisateur du client

Hôte Donne le nom d'hôte et le port du serveur auquel la requête est envoyée

Référent Fournit l'URL du document qui contient l'URI de la demande actuelle

UA-Color fournit des informations sur les capacités de couleur de l'écran de la machine cliente

UA-CPUc Indique le type ou le fabricant du processeur du client

UA-Disp Fournit des informations sur les capacités d'affichage (écran) du client

UA-OS donne le nom et la version du système d'exploitation exécuté sur la machine cliente

UA-Pixels fournit des informations sur les pixels de l'affichage de la machine cliente

User-Agent indique au serveur le nom de l'application effectuant la demande

a Les en-têtes Client-IP et UA-* ne sont pas définis dans la RFC 2616 mais sont implémentés par de nombreuses applications clientes HTTP. b Un format d'adresse e-mail RFC 822.

Bien qu'implémentés par certains clients, les en-têtes UA-* peuvent être considérés comme nuisibles. Le contenu, notamment HTML, ne doit pas être ciblé sur des configurations client spécifiques.

Accepter les en-têtes

Les en-têtes d'acceptation permettent au client d'indiquer aux serveurs ses préférences et ses capacités : ce qu'il souhaite, ce qu'il peut utiliser et, surtout, ce qu'il ne souhaite pas. Les serveurs peuvent ensuite utiliser ces informations supplémentaires pour prendre des décisions plus éclairées sur les messages à envoyer. Les en-têtes d'acceptation bénéficient aux deux côtés de la connexion. Les clients obtiennent ce qu'ils souhaitent, et les serveurs ne perdent pas de temps et de bande passante à envoyer des messages que le client ne peut pas utiliser. Le tableau 3-14 répertorie les différents en-têtes d'acceptation. Tableau 3-14. En-têtes d'acceptation

Description de l'en-tête

Accepter Indique au serveur quels types de médias peuvent être envoyés

Accept-Charset indique au serveur quels jeux de caractères peuvent être envoyés

Accept-Encoding indique au serveur quels encodages peuvent être envoyés

Accept-Language indique au serveur quelles langues peuvent être envoyées

TEa indique au serveur quels codages de transfert d'extension peuvent être utilisés

a Voir « En-têtes de codage de transfert » au chapitre 15 pour plus d'informations sur l'en-tête TE.

En-têtes de requête conditionnelle

Parfois, les clients souhaitent imposer des restrictions à une requête. Par exemple, si le client possède déjà une copie d'un document, il peut demander au serveur de l'envoyer uniquement s'il est différent de la copie dont il dispose déjà. Grâce aux en-têtes de requête conditionnels, les clients peuvent imposer de telles restrictions, obligeant le serveur à vérifier que les conditions sont remplies avant de satisfaire la requête. Le tableau 3-15 répertorie les différents en-têtes de requête conditionnels. Tableau 3-15. En-têtes de requête conditionnels

Description de l'en-tête

Expect permet à un client de lister les comportements de serveur dont il a besoin pour une requête

If-Match Obtient le document si la balise d'entité correspond à la balise d'entité actuelle pour le documenta

If-Modified-Since Restreint la demande sauf si la ressource a été modifiée depuis la date spécifiée

If-None-Match Obtient le document si les balises d'entité fournies ne correspondent pas à celles du document actuel

If-Range Permet une demande conditionnelle pour une plage d'un document

If-Unmodified-Since Restreint la demande à moins que la ressource n'ait pas été modifiée depuis la date spécifiée

Demande de plage Demande une plage spécifique d'une ressource, si le serveur prend en charge les demandes de plage

a Voir le chapitre 7 pour plus d'informations sur les balises d'entité. La balise est essentiellement un identifiant pour une version de la ressource. b Voir « Requêtes de plage » au chapitre 15 pour plus d'informations sur l'en-tête Plage.

Demander des en-têtes de sécurité

HTTP prend en charge nativement un schéma d'authentification simple par défi/réponse pour les requêtes. Ce schéma vise à sécuriser

légèrement les transactions en exigeant que les clients s'authentifient avant d'accéder à certaines ressources. Nous abordons ce schéma par défi/réponse au chapitre 14, ainsi que d'autres schémas de sécurité implémentés sur HTTP. Le tableau 3-16 répertorie les en-têtes de sécurité des requêtes. Tableau 3-16. En-têtes de sécurité des requêtes

Description de l'en-tête

Autorisation Contient les données que le client fournit au serveur pour s'authentifier

Cookie utilisé par les clients pour transmettre un jeton au serveur. Il ne s'agit pas d'un véritable en-tête de sécurité, mais il a des implications en matière de sécurité.

Cookie2 Utilisé pour noter la version des cookies prise en charge par un demandeur ; voir « Cookies version 1 (RFC 2965) » au chapitre 11

a L'en-tête Cookie n'est pas défini dans la RFC 2616 ; il est décrit en détail au chapitre 11.

En-têtes de requête proxy

Les proxys devenant de plus en plus courants sur Internet, quelques en-têtes ont été définis pour améliorer leur fonctionnement. Le chapitre 6 détaille ces en-têtes. Le tableau 3-17 répertorie les en-têtes de requêtes proxy.

Tableau 3-17. En-têtes de requête proxy

Description de l'en-tête

Max-Forwards Le nombre maximal de fois qu'une requête doit être transmise à un autre proxy ou passerelle sur son chemin vers le serveur d'origine - utilisé avec la méthode TRACE

Autorisation par proxy Identique à l'autorisation, mais utilisé lors de l'authentification avec un proxy

Connexion proxy Identique à Connexion, mais utilisé lors de l'établissement de connexions avec un proxy

a Voir « Max-Forwards » au chapitre 6.

En-têtes de réponse

Les messages de réponse possèdent leurs propres en-têtes. Ces en-têtes fournissent aux clients des informations supplémentaires, telles que l'expéditeur, les capacités du répondeur ou même des instructions spécifiques concernant la réponse. Ces en-têtes aident le client à gérer la réponse et à formuler des requêtes plus pertinentes. Le tableau 3-18 répertorie les en-têtes d'information des réponses. Tableau 3-18. En-têtes d'information des réponses

Description de l'en-tête

Âge Quel âge a la réponse

Publicb Une liste de méthodes de requête que le serveur prend en charge pour ses ressources

Réessayer après Une date ou une heure pour réessayer, si une ressource n'est pas disponible

Serveur Le nom et la version du logiciel d'application du serveur

Titrec Pour les documents HTML, le titre tel que donné par la source du document HTML

Avertissement Un message d'avertissement plus détaillé que celui figurant dans la phrase de raison

a Implique que la réponse a transité par un intermédiaire, éventuellement à partir d'un cache proxy. b L'en-tête Public est défini dans la RFC 2068 mais n'apparaît pas dans la dernière définition HTTP (RFC 2616). c L'en-tête Title n'est pas défini dans la RFC 2616 ; voir le projet de définition HTTP/1.0 d'origine (http://www.w3.org/Protocols/HTTP/HTTP2.html).

En-têtes de négociation

HTTP/1.1 permet aux serveurs et aux clients de négocier une ressource si plusieurs représentations sont disponibles, par exemple lorsqu'un document HTML est traduit en français et en allemand sur un serveur. Le chapitre 17 détaille la négociation. Voici quelques en-têtes utilisés par les serveurs pour transmettre des informations sur les ressources négociables. Le tableau 3-19 répertorie les en-têtes de négociation.

Tableau 3-19. En-têtes de négociation

Description de l'en-tête

Accept-Ranges Le type de plages qu'un serveur acceptera pour cette ressource

Varier Une liste d'autres en-têtes que le serveur examine et qui peuvent entraîner une variation de la réponse ; c'est-à-dire une liste d'en-têtes que le serveur examine pour choisir la meilleure version d'une ressource à envoyer au client

En-têtes de sécurité de réponse

Vous avez déjà vu les en-têtes de sécurité des requêtes, qui constituent le côté réponse du schéma d'authentification HTTP par défi/réponse. Nous détaillerons la sécurité au chapitre 14. Pour l'instant, voici les en-têtes de défi de base. Le tableau 3-20 répertorie les en-têtes de sécurité des réponses.

Tableau 3-20. En-têtes de sécurité des réponses

Description de l'en-tête

Proxy-Authenticate Une liste de défis pour le client du proxy

Set-Cookie Ce n'est pas un véritable en-tête de sécurité, mais il a des implications en matière de sécurité ; il est utilisé pour définir un jeton côté client que le serveur peut utiliser pour identifier le client.

Set-Cookie2 Similaire à Set-Cookie, définition de cookie RFC 2965 ; voir « Cookies version 1 (RFC 2965) » au chapitre 11

WWW-Authenticate Une liste de défis pour le client du serveur

a Set-Cookie et Set-Cookie2 sont des en-têtes d'extension qui sont également traités dans le chapitre 11.

En-têtes d'entité

Il existe de nombreux en-têtes pour décrire la charge utile des messages HTTP. Étant donné que les messages de requête et de réponse peuvent contenir des entités, ces en-têtes peuvent apparaître dans les deux types de messages.

Les en-têtes d'entité fournissent un large éventail d'informations sur l'entité et son contenu, allant du type d'objet aux méthodes de requête valides applicables à la ressource. En général, ils indiquent au destinataire du message à quoi il a affaire. Le tableau 3-21 répertorie les en-têtes d'information sur l'entité.

Tableau 3-21. En-têtes d'information des entités

Description de l'en-tête

Autoriser Répertorie les méthodes de requête qui peuvent être exécutées sur cette entité

Emplacement Indique au client où se trouve réellement l'entité ; utilisé pour diriger le récepteur vers un emplacement (éventuellement nouveau) (URL) pour la ressource

En-têtes de contenu

Les en-têtes de contenu fournissent des informations spécifiques sur le contenu de l'entité, révélant son type, sa taille et d'autres informations utiles à son traitement. Par exemple, un navigateur web peut analyser le type de contenu renvoyé et savoir comment afficher l'objet. Le tableau 3-22 répertorie les différents en-têtes de contenu.

Tableau 3-22. En-têtes de contenu

Description de l'en-tête

Content-Basea L'URL de base pour résoudre les URL relatives dans le corps

Codage de contenu Tout codage effectué sur le corps

Tableau 3-22. En-têtes de contenu (suite)

Description de l'en-tête

Contenu-Langue Le langage naturel le mieux utilisé pour comprendre le corps

Contenu-Longueur La longueur ou la taille du corps

Emplacement du contenu Où se trouve réellement la ressource

Contenu-MD5 Une somme de contrôle MD5 du corps

Plage de contenu La plage d'octets que cette entité représente à partir de la ressource entière

Type de contenu Le type d'objet que ce corps est

a L'en-tête Content-Base n'est pas défini dans la RFC 2616.

En-têtes de mise en cache d'entités

Les en-têtes de mise en cache généraux fournissent des directives sur la manière et le moment de la mise en cache. Les en-têtes de mise en cache d'entité fournissent des informations sur l'entité mise en cache, par exemple, les informations nécessaires pour vérifier la validité d'une copie de la ressource en cache et des conseils pour mieux estimer le moment où une ressource en cache risque de ne plus être valide.

Dans le chapitre 7, nous plongeons au cœur de la mise en cache des requêtes et des réponses HTTP.

Nous reverrons ces en-têtes ici. Le tableau 3-23 répertorie les en-têtes de mise en cache des entités.

Tableau 3-23. En-têtes de mise en cache des entités.

Description de l'en-tête

ETag La balise d'entité associée à cette entité

Expire La date et l'heure à laquelle cette entité ne sera plus valide et devra être récupérée à partir de la source d'origine

Dernière modification La dernière date et heure à laquelle cette entité a été modifiée

Les balises d’entité sont essentiellement des identifiants pour une version particulière d’une ressource.

Pour plus d'informations

Pour plus d'informations, reportez-vous à :

http://www.w3.org/Protocols/rfc2616/rfc2616.txt

RFC 2616, « Protocole de transfert hypertexte », par R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Mastinter, P. Leach et T. Berners-Lee.

Référence HTTP Pocket

Clintin Wong, O'Reilly & Associates, Inc.

http://www.w3.org/Protocols/

La page d'architecture W3C pour HTTP.

Pour plus d'informations

Chapitre 4Ceci est le titre du livreCHAPITRE 4

Gestion des connexions

Les spécifications HTTP expliquent assez bien les messages HTTP, mais elles ne parlent pas beaucoup des connexions HTTP, le réseau critique par lequel transitent les messages HTTP. Si vous êtes programmeur et développez des applications HTTP, vous devez comprendre les tenants et aboutissants des connexions HTTP et comment les utiliser.

La gestion des connexions HTTP est une sorte de science obscure, apprise autant par l'expérimentation et l'apprentissage que par la littérature. Dans ce chapitre, vous découvrirez :

• Comment HTTP utilise les connexions TCP

• Retards, goulots d'étranglement et obstructions dans les connexions TCP

• Optimisations HTTP, y compris les connexions parallèles, persistantes et en pipeline

• À faire et à ne pas faire pour la gestion des connexions

Connexions TCP

La quasi-totalité des communications HTTP mondiales s'effectue via TCP/IP, un ensemble de protocoles réseau à commutation de paquets en couches répandu, utilisé par les ordinateurs et les périphériques réseau du monde entier. Une application cliente peut établir une connexion TCP/IP avec une application serveur, fonctionnant pratiquement partout dans le monde. Une fois la connexion établie, les messages échangés entre les ordinateurs client et serveur ne seront jamais perdus, endommagés ou reçus dans le désordre.*

Supposons que vous souhaitiez connaître la dernière liste de prix des outils électriques du magasin Joe's Hardware :

http://www.joes-hardware.com:80/power-tools.html

Une fois cette URL reçue, votre navigateur exécute les étapes illustrées à la figure 4-1. Aux étapes 1 à 3, l'adresse IP et le numéro de port du serveur sont extraits de l'URL. Un protocole TCP

* Bien que les messages ne soient ni perdus ni corrompus, la communication entre le client et le serveur peut être interrompue en cas de panne d'un ordinateur ou d'un réseau. Dans ce cas, le client et le serveur sont informés de la rupture de communication.

La connexion au serveur Web est établie à l'étape 4 et un message de requête est envoyé via la connexion à l'étape 5. La réponse est lue à l'étape 6 et la connexion est fermée à l'étape 7.

Figure 4-1. Les navigateurs Web communiquent avec les serveurs Web via des connexions TCP

Canaux de données TCP fiables

Les connexions HTTP ne sont rien d'autre que des connexions TCP, avec quelques règles d'utilisation. Les connexions TCP sont les connexions fiables d'Internet. Pour envoyer des données avec précision et rapidité, vous devez connaître les bases de TCP.*

TCP fournit à HTTP un canal de bits fiable. Les octets insérés d'un côté d'une connexion TCP ressortent correctement de l'autre côté, dans le bon ordre (voir figure 4-2).

* Si vous souhaitez développer des applications HTTP sophistiquées, et surtout si vous souhaitez qu'elles soient rapides, vous souhaiterez en apprendre davantage sur le fonctionnement et les performances de TCP que ce que nous abordons dans ce chapitre. Nous vous recommandons les livres « TCP/IP Illustrated » de W. Richard Stevens (Addison Wesley).

Figure 4-2. TCP transporte les données HTTP dans l'ordre et sans corruption

Les flux TCP sont segmentés et expédiés par paquets IP

TCP envoie ses données par petits blocs appelés paquets IP (ou datagrammes IP). Ainsi, HTTP constitue la couche supérieure d'une « pile de protocoles » de type « HTTP sur TCP sur IP », comme illustré à la figure 4-3a. Une variante sécurisée, HTTPS, insère une couche de chiffrement cryptographique (appelée TLS ou SSL) entre HTTP et TCP (figure 4-3b).

Figure 4-3. Piles de protocoles réseau HTTP et HTTPS

Lorsque HTTP souhaite transmettre un message, il diffuse le contenu du message, dans l'ordre, via une connexion TCP ouverte. TCP prend le flux de données, le découpe en segments, et transporte ces segments

sur Internet dans des enveloppes appelées paquets IP (voir Figure 4-4). Tout cela est géré par le logiciel TCP/IP ; le programmeur HTTP n'en voit rien.

Chaque segment TCP est transporté par un paquet IP d'une adresse IP à une autre. Chacun de ces paquets IP contient :

• Un en-tête de paquet IP (généralement 20 octets)

• Un en-tête de segment TCP (généralement 20 octets)

• Un bloc de données TCP (0 octet ou plus)

L'en-tête IP contient les adresses IP source et de destination, la taille et d'autres indicateurs. L'en-tête de segment TCP contient les numéros de port TCP, les indicateurs de contrôle TCP et les valeurs numériques utilisées pour l'ordonnancement des données et le contrôle d'intégrité.

Figure 4-4. Les paquets IP transportent des segments TCP, qui transportent des morceaux du flux de données TCP

Maintenir les connexions TCP droites

Un ordinateur peut avoir plusieurs connexions TCP ouvertes simultanément. TCP maintient toutes ces connexions en ligne via les numéros de port.

Les numéros de port sont comme les extensions téléphoniques des employés. Tout comme le numéro de téléphone principal d'une entreprise vous permet d'accéder à la réception et l'extension vous permet d'accéder au bon employé, l'adresse IP vous permet d'accéder au bon ordinateur et le numéro de port à la bonne application. Une connexion TCP se distingue par quatre valeurs :

<adresse IP source, port source, adresse IP de destination, port de destination>

Ensemble, ces quatre valeurs définissent une connexion de manière unique. Deux connexions TCP différentes ne peuvent pas avoir les mêmes valeurs pour les quatre composants d'adresse (mais des connexions différentes peuvent avoir les mêmes valeurs pour certains composants).

Dans la Figure 4-5, il y a quatre connexions : A, B, C et D. Les informations pertinentes pour chaque port sont répertoriées dans le Tableau 4-1.

Tableau 4-1. Valeurs de connexion TCP

Connexion Adresse IP source Port source Adresse IP de destination Port de destination

A 209.1.32.34 2034 204.62.128.58 4133

B 209.1.32.35 3227 204.62.128.58 4140

C 209.1.32.35 3105 207.25.71.25 80

D 209.1.33.89 5100 207.25.71.25 80

Figure 4-5. Quatre connexions TCP distinctes

Notez que certaines connexions partagent le même numéro de port de destination (C et D ont toutes deux le port de destination 80). Certaines connexions ont la même adresse IP source (B et C). D'autres ont la même adresse IP de destination (A et B, et C et

D). Mais aucune connexion différente ne partage les quatre valeurs identiques.

Programmation avec les sockets TCP

Les systèmes d'exploitation offrent différentes fonctionnalités pour manipuler leurs connexions TCP. Examinons rapidement une interface de programmation TCP pour illustrer les choses. Le tableau 4-2 présente quelques-unes des principales interfaces fournies par l'API sockets. Cette API cache tous les détails de TCP et IP au programmeur HTTP. Initialement développée pour le système d'exploitation Unix, l'API sockets propose désormais des variantes pour presque tous les systèmes d'exploitation et langages.

Tableau 4-2. Fonctions courantes des interfaces de socket pour la programmation des connexions TCP

Description de l'appel API Sockets

s = socket(<paramètres>) Crée un nouveau socket sans nom et non attaché. bind(s, <IP locale:port>) Attribue un numéro de port local et une interface au socket.

Tableau 4-2. Fonctions courantes de l'interface socket pour la programmation des connexions TCP (suite)

Description de l'appel API Sockets

connect(s, <IP distante : port>) Établit une connexion TCP à un socket local et à un hôte et un port distants.

listen(s,...) Marque un socket local comme légal pour accepter les connexions.

s2 = accept(s) Attend que quelqu'un établisse une connexion à un port local.

n = read(s,buffer,n) Tente de lire n octets du socket dans le tampon.

n = write(s,buffer,n) Tente d'écrire n octets du tampon dans le socket.

close(s) Ferme complètement la connexion TCP.

shutdown(s,<side>) Ferme uniquement l'entrée ou la sortie de la connexion TCP.

getsockopt(s, ...) Lit la valeur d'une option de configuration de socket interne.

setsockopt(s, ...) Modifie la valeur d'une option de configuration de socket interne.

L'API sockets vous permet de créer des structures de données pour les points de terminaison TCP, de les connecter aux points de terminaison TCP des serveurs distants et de lire et écrire des flux de données. L'API TCP masque tous les détails de la négociation du protocole réseau sous-jacent, ainsi que la segmentation et le réassemblage du flux de données TCP vers et depuis les paquets IP.

Dans la figure 4-1, nous avons montré comment un navigateur web pouvait télécharger la page power-tools.html depuis la quincaillerie Joe's via HTTP. Le pseudo-code de la figure 4-6 illustre l'utilisation de l'API sockets pour illustrer les étapes que le client et le serveur pourraient suivre pour implémenter cette transaction HTTP.

Figure 4-6. Communication entre clients et serveurs TCP via l'interface sockets TCP

Le serveur web attend une connexion (Figure 4-6, S4). Le client détermine l'adresse IP et le numéro de port à partir de l'URL et établit une connexion TCP avec le serveur (Figure 4-6, C3). L'établissement d'une connexion peut prendre un certain temps, selon la distance du serveur, sa charge et la congestion d'Internet.

Une fois la connexion établie, le client envoie la requête HTTP (Figure 4-6, C5) et le serveur la lit (Figure 4-6, S6). Une fois le message de requête reçu, le serveur le traite, exécute l'action demandée (Figure 4-6, S7) et réécrit les données au client. Le client lit la requête (Figure 4-6, C6) et traite les données de réponse (Figure 4-6, C7).

Considérations sur les performances TCP

Comme HTTP repose directement sur TCP, les performances des transactions HTTP dépendent fortement de celles du réseau TCP sous-jacent. Cette section met en évidence quelques points importants concernant les performances de ces connexions TCP. En comprenant certaines des caractéristiques de performance de base de TCP, vous comprendrez mieux les fonctionnalités d'optimisation des connexions de HTTP et serez en mesure de concevoir et de mettre en œuvre des applications HTTP plus performantes.

Cette section nécessite une compréhension des détails internes du protocole TCP. Si les considérations relatives aux performances TCP ne vous intéressent pas (ou ne vous y connaissent pas), n'hésitez pas à passer directement à la section « Gestion des connexions HTTP ». TCP étant un sujet complexe, nous ne pouvons ici fournir qu'un bref aperçu de ses performances. Consultez la section « Pour plus d'informations » à la fin de ce chapitre pour une liste d'excellentes références TCP.

Retards de transaction HTTP

Commençons notre présentation des performances TCP en examinant les délais réseau qui surviennent lors d'une requête HTTP. La figure 4-7 illustre les principaux délais de connexion, de transfert et de traitement d'une transaction HTTP.

Figure 4-7. Chronologie d'une transaction HTTP série

Notez que le temps de traitement des transactions peut être relativement court comparé au temps nécessaire à l'établissement des connexions TCP et au transfert des messages de requête et de réponse. À moins que le client ou le serveur ne soit surchargé ou n'exécute des ressources dynamiques complexes, la plupart des retards HTTP sont dus à des retards du réseau TCP.

Il existe plusieurs causes possibles de retard dans une transaction HTTP :

1. Un client doit d'abord déterminer l'adresse IP et le numéro de port du serveur web à partir de l'URI. Si le nom d'hôte de l'URI n'a pas été visité récemment, la conversion du nom d'hôte de l'URI en adresse IP via l'infrastructure de résolution DNS peut prendre plusieurs dizaines de secondes.*

2. Ensuite, le client envoie une requête de connexion TCP au serveur et attend que celui-ci lui renvoie une réponse d'acceptation. Un délai d'établissement de connexion se produit à chaque nouvelle connexion TCP. Ce délai ne dure généralement qu'une ou deux secondes, mais il peut s'allonger rapidement lorsque des centaines de transactions HTTP sont effectuées.

3. Une fois la connexion établie, le client envoie la requête HTTP via le canal TCP nouvellement établi. Le serveur web lit le message de requête de la connexion TCP à mesure que les données arrivent et traite la requête. Le message de requête met du temps à circuler sur Internet et à être traité par le serveur.

4. Le serveur Web réécrit ensuite la réponse HTTP, ce qui prend également du temps.

L'ampleur de ces retards sur le réseau TCP dépend de la vitesse du matériel, de la charge du réseau et du serveur, de la taille des messages de requête et de réponse, ainsi que de la distance entre le client et le serveur. Ces retards sont également fortement influencés par les complexités techniques du protocole TCP.

Domaines d'intérêt en matière de performance

Le reste de cette section décrit certains des retards les plus courants liés à TCP affectant les programmeurs HTTP, y compris les causes et les impacts sur les performances de :

• La poignée de main de configuration de la connexion TCP

• Contrôle de congestion à démarrage lent TCP

• L'algorithme de Nagle pour l'agrégation de données

• Algorithme d'accusé de réception différé de TCP pour les accusés de réception superposés

• Délais TIME_WAIT et épuisement des ports

Si vous développez un logiciel HTTP hautes performances, vous devez comprendre chacun de ces facteurs. Si vous n'avez pas besoin de ce niveau d'optimisation des performances, n'hésitez pas à passer à la section suivante.

* Heureusement, la plupart des clients HTTP conservent un petit cache DNS contenant les adresses IP des sites récemment consultés. Lorsque l'adresse IP est déjà « mise en cache » (enregistrée) localement, la recherche est instantanée. La navigation web étant principalement axée sur un nombre limité de sites populaires, les noms d'hôtes sont généralement résolus très rapidement.

Délais de négociation de connexion TCP

Lors de la configuration d'une nouvelle connexion TCP, avant même l'envoi de données, le logiciel TCP échange une série de paquets IP pour négocier les conditions de la connexion (voir Figure 4-8). Ces échanges peuvent dégrader considérablement les performances HTTP si les connexions sont utilisées pour des transferts de données de faible ampleur.

Figure 4-8. TCP nécessite deux transferts de paquets pour établir la connexion avant de pouvoir envoyer des données.

Voici les étapes de la négociation de la connexion TCP :

1. Pour demander une nouvelle connexion TCP, le client envoie un petit paquet TCP (généralement de 40 à 60 octets) au serveur. Ce paquet possède un indicateur « SYN » spécifique, indiquant qu'il s'agit d'une demande de connexion. Ceci est illustré à la figure 4-8a.

2. Si le serveur accepte la connexion, il calcule certains paramètres de connexion et renvoie un paquet TCP au client, avec les indicateurs « SYN » et « ACK » définis, indiquant que la demande de connexion est acceptée (voir Figure 4-8b).

3. Enfin, le client envoie un accusé de réception au serveur, l'informant que la connexion a été établie avec succès (voir Figure 4-8c). Les piles TCP modernes permettent au client d'envoyer des données dans ce paquet d'accusé de réception.

Le programmeur HTTP ne voit jamais ces paquets ; ils sont gérés de manière invisible par le logiciel TCP/IP. Il ne voit qu'un délai lors de la création d'une nouvelle connexion TCP.

L'établissement de liaison SYN/SYN+ACK (Figure 4-8a et b) crée un délai mesurable lorsque les transactions HTTP n'échangent que peu de données, comme c'est souvent le cas. Le paquet ACK de connexion TCP (Figure 4-8c) est souvent suffisamment volumineux pour contenir l'intégralité du message de requête HTTP*, et de nombreux messages de réponse du serveur HTTP tiennent dans un seul paquet IP (par exemple, lorsqu'il s'agit d'un petit fichier HTML d'un élément graphique décoratif ou d'une réponse 304 « Non modifié » à une requête de cache de navigateur).

* Les paquets IP font généralement quelques centaines d’octets pour le trafic Internet et environ 1 500 octets pour le trafic local.

Au final, les petites transactions HTTP peuvent consacrer 50 % ou plus de leur temps à la configuration TCP. Les sections suivantes expliqueront comment HTTP permet la réutilisation des connexions existantes afin d'éliminer l'impact de ce délai de configuration TCP.

Accusés de réception différés

Étant donné qu'Internet lui-même ne garantit pas une livraison fiable des paquets (les routeurs Internet sont libres de détruire les paquets à volonté s'ils sont surchargés), TCP implémente son propre système d'accusé de réception pour garantir une livraison réussie des données.

Chaque segment TCP reçoit un numéro de séquence et une somme de contrôle d'intégrité des données. Le récepteur de chaque segment renvoie de petits paquets d'accusé de réception à l'expéditeur lorsque les segments ont été reçus intacts. Si l'expéditeur ne reçoit pas d'accusé de réception dans un délai spécifié, il conclut que le paquet a été détruit ou corrompu et renvoie les données.

Comme les accusés de réception sont courts, TCP leur permet de se greffer sur les paquets de données sortants allant dans la même direction. En combinant les accusés de réception renvoyés avec les paquets de données sortants, TCP optimise l'utilisation du réseau. Pour augmenter les chances qu'un accusé de réception trouve un paquet de données allant dans la même direction, de nombreuses piles TCP

implémentent un algorithme d'« accusé de réception différé ». Les accusés de réception différés conservent les accusés de réception sortants dans une mémoire tampon pendant une certaine fenêtre de temps (généralement 100 à 200 millisecondes), en attendant un paquet de données sortant sur lequel se greffer. Si aucun paquet de données sortant n'arrive dans ce délai, l'accusé de réception est envoyé dans son propre paquet.

Malheureusement, le comportement bimodal requête-réponse de HTTP réduit les risques de piggybacking. Il y a peu de paquets qui repartent en sens inverse au moment souhaité. La désactivation des algorithmes d'accusé de réception entraîne souvent des retards importants. Selon votre système d'exploitation, vous pouvez ajuster ou désactiver l'algorithme d'accusé de réception différé.

Avant de modifier les paramètres de votre pile TCP, assurez-vous de bien maîtriser votre activité. Les algorithmes de TCP ont été introduits pour protéger Internet des applications mal conçues. Si vous modifiez des configurations TCP, assurez-vous absolument que votre application ne créera pas les problèmes que les algorithmes étaient censés éviter.

Démarrage lent TCP

Les performances du transfert de données TCP dépendent également de l'ancienneté de la connexion. Les connexions TCP s'ajustent au fil du temps, limitant initialement leur vitesse maximale et l'augmentant progressivement à mesure que les données sont transmises. Ce réglage, appelé démarrage lent TCP, permet d'éviter les surcharges et congestions soudaines d'Internet.

Le démarrage lent TCP limite le nombre de paquets qu'un point de terminaison TCP peut recevoir simultanément. En termes simples, chaque fois qu'un paquet est reçu avec succès, l'expéditeur est autorisé à en envoyer deux autres. Si une transaction HTTP a une grande quantité de données à envoyer, elle ne peut pas envoyer tous les paquets simultanément. Elle doit envoyer un paquet et attendre un accusé de réception ; elle peut ensuite envoyer deux paquets, chacun devant être acquitté, ce qui autorise quatre paquets, etc. C'est ce qu'on appelle « ouvrir la fenêtre de congestion ».

Grâce à cette fonctionnalité de contrôle de congestion, les nouvelles connexions sont plus lentes que les connexions « réglées » ayant déjà échangé une quantité modeste de données. Comme les connexions réglées sont plus rapides, HTTP inclut des fonctionnalités permettant de réutiliser les connexions existantes. Nous aborderons ces « connexions persistantes » HTTP plus loin dans ce chapitre.

Algorithme de Nagle et TCP_NODELAY

TCP dispose d'une interface de flux de données qui permet aux applications de diffuser des données de toute taille vers la pile TCP,

même un seul octet à la fois ! Cependant, comme chaque segment TCP contient au moins 40 octets d'indicateurs et d'en-têtes, les performances du réseau peuvent être fortement dégradées si TCP envoie un grand nombre de paquets contenant de petites quantités de données.*

L'algorithme de Nagle (du nom de son créateur, John Nagle) vise à regrouper une grande quantité de données TCP avant l'envoi d'un paquet, améliorant ainsi l'efficacité du réseau. Cet algorithme est décrit dans la RFC 896, « Contrôle de congestion dans les réseaux IP/TCP ».

L'algorithme de Nagle décourage l'envoi de segments de taille inférieure (la taille maximale d'un paquet est d'environ 1 500 octets sur un réseau local, ou de quelques centaines d'octets sur Internet). Il permet d'envoyer un paquet de taille inférieure uniquement si tous les autres paquets ont été acquittés. Si d'autres paquets sont encore en transit, les données partielles sont mises en mémoire tampon. Ces données mises en mémoire tampon ne sont envoyées que lorsque les paquets en attente sont acquittés ou lorsque la mémoire tampon a accumulé suffisamment de données pour envoyer un paquet complet†.

L'algorithme de Nagle engendre plusieurs problèmes de performances HTTP. Premièrement, les petits messages HTTP peuvent ne pas remplir un paquet, ce qui peut les retarder en attendant des données supplémentaires qui n'arriveront jamais. Deuxièmement, l'algorithme de Nagle interagit mal avec les accusés de réception désactivés : il retarde l'envoi des données jusqu'à l'arrivée d'un accusé de réception, mais l'accusé de réception lui-même est retardé de 100 à 200 millisecondes par l'algorithme d'accusé de réception différé.

Les applications HTTP désactivent souvent l'algorithme de Nagle pour améliorer les performances, en définissant le paramètre TCP_NODELAY sur leurs piles. Dans ce cas, veillez à écrire de gros blocs de données sur TCP afin d'éviter de créer une multitude de petits paquets.

* L'envoi d'une avalanche de paquets d'un seul octet est appelé « syndrome de la fenêtre idiote de l'expéditeur ». Ce phénomène est inefficace, antisocial et peut perturber le trafic Internet.

† Il existe plusieurs variantes de cet algorithme, notamment des délais d'attente et des modifications de la logique d'accusé de réception, mais l'algorithme de base provoque la mise en mémoire tampon de données plus petites qu'un segment TCP.

‡ Ces problèmes peuvent s’aggraver lors de l’utilisation de connexions pipelinées (décrites plus loin dans ce chapitre), car les clients peuvent avoir plusieurs messages à envoyer au même serveur et ne veulent pas de retards.

Accumulation de TIME_WAIT et épuisement des ports

L'épuisement du port TIME_WAIT est un problème de performances grave qui affecte les analyses comparatives, mais qui est relativement rare dans les déploiements réels. Il mérite une attention particulière, car la plupart des personnes impliquées dans les analyses comparatives rencontrent ce problème et obtiennent des performances inattendues.

Lorsqu'un point de terminaison TCP ferme une connexion TCP, il conserve en mémoire un petit bloc de contrôle enregistrant les adresses IP et les numéros de port de la connexion récemment fermée. Ces informations sont conservées pendant une courte durée, généralement environ deux fois la durée de vie maximale estimée du segment (appelée « 2MSL » ; souvent deux minutes*), afin d'éviter qu'une nouvelle connexion TCP avec les mêmes adresses et numéros de port ne soit créée pendant ce temps. Cela empêche l'injection accidentelle de paquets dupliqués provenant de la connexion précédente dans une nouvelle connexion ayant les mêmes adresses et numéros de port. En pratique, cet algorithme empêche la création, la fermeture et la recréation de deux connexions avec exactement les mêmes adresses IP et numéros de port en deux minutes.

Avec les routeurs actuels à haut débit, il est extrêmement improbable qu'un paquet dupliqué apparaisse sur un serveur quelques minutes après la fermeture d'une connexion. Certains systèmes d'exploitation définissent une valeur inférieure pour 2MSL, mais soyez prudent lorsque vous la remplacez. Les paquets sont dupliqués et les données TCP seront corrompues si un paquet dupliqué d'une connexion précédente est inséré dans un nouveau flux avec les mêmes valeurs de connexion.

Le délai de fermeture de connexion 2MSL n'est généralement pas un problème, mais il peut l'être lors d'un benchmark. Il est fréquent qu'un ou quelques ordinateurs de test se connectent à un système soumis à un benchmark, ce qui limite le nombre d'adresses IP clientes connectées au serveur. De plus, le serveur écoute généralement sur le port TCP par défaut de HTTP, le port 80. Ces circonstances limitent les combinaisons de valeurs de connexion disponibles, à un moment où la réutilisation des numéros de port est bloquée par TIME_WAIT.

Dans une situation pathologique avec un client et un serveur Web, des quatre valeurs qui composent une connexion TCP :

<source-IP-address, source-port, destination-IP-address, destination-port> trois d'entre eux sont fixes, seul le port source est libre de changer :

<IP client, port source, IP serveur, 80>

Chaque fois que le client se connecte au serveur, il obtient un nouveau port source afin de bénéficier d'une connexion unique. Cependant, comme le nombre de ports sources disponibles est limité (par exemple, 60 000) et qu'aucune connexion ne peut être réutilisée pendant

2 secondes MSL (par exemple, 120 secondes), le taux de connexion est limité à 60 000/120 = 500 transactions/s. Si vous conservez

* La valeur 2MSL de deux minutes est historique. Autrefois, lorsque les routeurs étaient beaucoup plus lents, on estimait qu'une copie d'un paquet pouvait rester en file d'attente sur Internet jusqu'à une minute avant d'être détruite. Aujourd'hui, la durée de vie maximale d'un segment est bien inférieure.

Si vous effectuez des optimisations et que votre serveur ne dépasse pas 500 transactions/s, assurez-vous de ne pas rencontrer d'épuisement de port TIME_WAIT. Vous pouvez résoudre ce problème en utilisant davantage de machines génératrices de charge client ou en veillant à ce que le client et le serveur utilisent plusieurs adresses IP virtuelles pour augmenter les combinaisons de connexions.

Même si vous ne rencontrez pas de problèmes d'épuisement de ports, soyez vigilant face à un grand nombre de connexions ouvertes ou de blocs de contrôle alloués à des connexions en état d'attente. Certains systèmes d'exploitation ralentissent considérablement lorsqu'ils sont nombreux.

Gestion des connexions HTTP

Les deux premières sections de ce chapitre ont présenté un aperçu des connexions TCP et de leurs implications en termes de performances. Pour en savoir plus sur les réseaux TCP, consultez les ressources répertoriées à la fin du chapitre.

Nous allons maintenant changer de sujet et revenir directement à HTTP. La suite de ce chapitre explique la technologie HTTP pour la manipulation et l'optimisation des connexions. Nous commencerons par l'en-tête de connexion HTTP, un élément souvent mal compris mais essentiel de la gestion des connexions HTTP. Nous aborderons ensuite les techniques d'optimisation des connexions HTTP.

L'en-tête de connexion souvent mal compris

HTTP permet une chaîne d'intermédiaires HTTP entre le client et le serveur d'origine (proxies, caches, etc.). Les messages HTTP sont transmis étape par étape du client, via des périphériques intermédiaires, jusqu'au serveur d'origine (ou inversement).

Dans certains cas, deux applications HTTP adjacentes peuvent souhaiter appliquer un ensemble d'options à leur connexion partagée. Le champ d'en-tête « Connexion HTTP » contient une liste de jetons de connexion, séparés par des virgules, spécifiant des options pour la connexion qui ne sont pas propagées aux autres connexions. Par exemple, une connexion devant être fermée après l'envoi du message suivant peut être indiquée par « Connexion : close ».

L'en-tête de connexion est parfois déroutant, car il peut transporter trois types de jetons différents :

• Noms de champs d'en-tête HTTP, répertoriant les en-têtes pertinents uniquement pour cette connexion

• Valeurs de jeton arbitraires, décrivant des options non standard pour cette connexion

• La valeur close, indiquant que la connexion persistante sera fermée une fois terminée

Si un jeton de connexion contient le nom d'un champ d'en-tête HTTP, ce champ contient des informations spécifiques à la connexion et ne doit pas être transféré. Tous les champs d'en-tête répertoriés dans l'en-tête de connexion doivent être supprimés avant la transmission du message. L'insertion d'un nom d'en-tête saut par saut dans un en-tête de connexion est appelée

« Protection de l'en-tête », car l'en-tête de connexion protège contre toute transmission accidentelle de l'en-tête local. Un exemple est présenté à la figure 4-9.

Figure 4-9. L'en-tête de connexion permet à l'expéditeur de spécifier des options de connexion spécifiques.

Lorsqu'une application HTTP reçoit un message contenant un en-tête de connexion, le récepteur analyse et applique toutes les options demandées par l'expéditeur. Il supprime ensuite l'en-tête de connexion et tous les en-têtes listés dans celui-ci avant de transmettre le message au saut suivant. De plus, certains en-têtes saut par saut peuvent ne pas être listés comme valeurs d'un en-tête de connexion, mais ne doivent pas être proxy. Il s'agit notamment de Proxy-Authenticate, Proxy-Connection, Transfer-Encoding et Upgrade. Pour plus d'informations sur l'en-tête de connexion, consultez l'annexe C.

Retards dans les transactions en série

Les retards de performances TCP peuvent s'accumuler si les connexions sont gérées de manière naïve. Par exemple, supposons que vous ayez une page web avec trois images intégrées. Votre navigateur doit émettre quatre transactions HTTP pour afficher cette page : une pour le code HTML principal et trois pour les images intégrées. Si chaque transaction nécessite une nouvelle connexion,

connexion, les délais de connexion et de démarrage lent peuvent s'accumuler (voir Figure 4-10).*

Figure 4-10. Quatre transactions (en série)

* Pour les besoins de cet exemple, supposons que tous les objets ont à peu près la même taille et sont hébergés sur le même serveur, et que l'entrée DNS est mise en cache, éliminant ainsi le temps de recherche DNS.

Gestion des connexions HTTP

En plus du délai réel imposé par le chargement en série, il existe également une perception psychologique de lenteur lorsqu'une seule image se charge et que rien ne se passe dessus.

le reste de la page. Les utilisateurs préfèrent que plusieurs images se chargent simultanément.*

Un autre inconvénient du chargement en série est que certains navigateurs ne peuvent rien afficher à l'écran tant que suffisamment d'objets ne sont pas chargés, car ils ne connaissent pas la taille des objets avant leur chargement, et ils peuvent avoir besoin de ces informations pour déterminer leur positionnement à l'écran. Dans ce cas, le navigateur peut bien progresser dans le chargement en série des objets, mais l'utilisateur peut se retrouver face à un écran blanc, ignorant toute progression.†

Plusieurs techniques, actuelles et émergentes, permettent d'améliorer les performances des connexions HTTP. Les sections suivantes présentent quatre de ces techniques :

Connexions parallèles

Requêtes HTTP simultanées sur plusieurs connexions TCP

Connexions persistantes

Réutilisation des connexions TCP pour éliminer les délais de connexion/fermeture

Connexions par pipeline

Requêtes HTTP simultanées sur une connexion TCP partagée

Connexions multiplexées

Entrelacement de morceaux de requêtes et de réponses (expérimental)

Connexions parallèles

Comme nous l'avons mentionné précédemment, un navigateur pourrait naïvement traiter chaque objet intégré en série en demandant complètement la page HTML d'origine, puis le premier objet intégré, puis le deuxième objet intégré, etc. Mais c'est trop lent !

HTTP permet aux clients d'ouvrir plusieurs connexions et d'effectuer plusieurs transactions HTTP en parallèle, comme illustré à la figure 4-

11. Dans cet exemple, quatre images intégrées sont chargées en parallèle, chaque transaction disposant de sa propre connexion TCP.

Les connexions parallèles peuvent accélérer le chargement des pages

Les pages composites constituées d'objets intégrés peuvent se charger plus rapidement si elles exploitent les temps morts et les limites de bande passante d'une connexion unique. Ces retards peuvent être

* Ceci est vrai même si le chargement de plusieurs images en même temps est plus lent que le chargement d'images une par une !

Les utilisateurs perçoivent souvent le chargement de plusieurs images comme plus rapide.

† Les concepteurs HTML peuvent éliminer ce « délai de mise en page » en ajoutant explicitement des attributs de largeur et de hauteur aux balises HTML pour les objets intégrés tels que les images. Fournir explicitement la largeur et la hauteur de l'image intégrée permet au navigateur de prendre des décisions de mise en page graphique avant de recevoir les objets du serveur.

‡ Les composants intégrés n'ont pas tous besoin d'être hébergés sur le même serveur Web, de sorte que les connexions parallèles peuvent être établies vers plusieurs serveurs.

Figure 4-11. Chaque composant d'une page implique une transaction HTTP distincte.

se chevauchent, et si une seule connexion ne sature pas la bande passante Internet du client, la bande passante inutilisée peut être allouée au chargement d'objets supplémentaires.

La figure 4-12 présente une chronologie des connexions parallèles, nettement plus rapide que la figure 4-10. La page HTML est chargée en premier, puis les trois transactions restantes sont traitées simultanément, chacune avec sa propre connexion.* Comme les images sont chargées en parallèle, les délais de connexion se chevauchent.

Figure 4-12. Quatre transactions (parallèles)

Les connexions parallèles ne sont pas toujours plus rapides

Même si les connexions parallèles peuvent être plus rapides, elles ne le sont pas toujours. Lorsque la bande passante du réseau du client est limitée (par exemple, un navigateur connecté à

* Il y aura généralement toujours un petit délai entre chaque demande de connexion en raison des frais généraux du logiciel, mais les

demandes de connexion et les temps de transfert se chevauchent généralement.

Connexions parallèles

(Internet via un modem 28,8 Kbit/s), la majeure partie du temps peut être consacrée au simple transfert de données. Dans ce cas, une seule transaction HTTP vers un serveur rapide peut facilement consommer toute la bande passante disponible du modem. Si plusieurs objets sont chargés en parallèle, ils se disputeront cette bande passante limitée, ce qui ralentira proportionnellement le chargement, ce qui n'apportera que peu, voire aucun avantage en termes de performances.*

De plus, un grand nombre de connexions ouvertes peut consommer beaucoup de mémoire et engendrer des problèmes de performances. Les pages web complexes peuvent contenir des dizaines, voire des centaines, d'objets intégrés. Les clients peuvent ouvrir des centaines de connexions, mais peu de serveurs web le souhaitent, car ils traitent souvent simultanément les requêtes de nombreux autres utilisateurs. Une centaine d'utilisateurs simultanés, ouvrant chacun 100 connexions, surchargent le serveur de 10 000 connexions, ce qui peut entraîner un ralentissement important du serveur. Il en va de même pour les proxys à forte charge.

En pratique, les navigateurs utilisent des connexions parallèles, mais ils limitent leur nombre total à un nombre restreint (souvent quatre). Les serveurs sont libres de fermer les connexions excessives d'un client particulier.

Les connexions parallèles peuvent sembler plus rapides

D'accord, les connexions parallèles n'accélèrent pas toujours le chargement des pages. Mais même si elles n'accélèrent pas réellement le transfert des pages, comme nous l'avons dit précédemment, elles donnent souvent l'impression aux utilisateurs que la page se charge plus vite, car ils peuvent constater la progression de l'affichage parallèle de plusieurs composants à l'écran.† Les humains perçoivent que les pages web se chargent plus vite s'il y a beaucoup d'action à l'écran, même si un chronomètre indique en réalité que le temps de téléchargement total est plus lent !

Connexions persistantes

Les clients web ouvrent souvent des connexions vers le même site. Par exemple, la plupart des images intégrées à une page web proviennent souvent du même site, et un nombre important d'hyperliens pointent vers le même site. Ainsi, une application qui lance une requête HTTP vers un serveur enverra probablement d'autres requêtes à ce serveur dans un avenir proche (pour récupérer les images intégrées, par exemple). Cette propriété est appelée localité du site.

Pour cette raison, HTTP/1.1 (et les versions améliorées de HTTP/1.0) permet aux périphériques HTTP de conserver les connexions TCP ouvertes après la fin des transactions et de réutiliser les connexions préexistantes pour les futures requêtes HTTP. Les connexions TCP conservées

* En fait, en raison de la surcharge supplémentaire due aux connexions multiples, il est tout à fait possible que les connexions parallèles prennent plus de temps à charger la page entière que les téléchargements en série.

† Cet effet est amplifié par l'utilisation croissante d'images progressives qui produisent d'abord des approximations d'images à faible résolution et augmentent progressivement la résolution.

Les connexions ouvertes après la fin des transactions sont appelées connexions persistantes. Les connexions non persistantes sont fermées après chaque transaction. Les connexions persistantes restent ouvertes pendant toute la durée des transactions, jusqu'à ce que le client ou le serveur décide de les fermer.

En réutilisant une connexion inactive et persistante déjà ouverte vers le serveur cible, vous pouvez éviter la lenteur de la configuration. De plus, la connexion déjà ouverte évite la phase d'adaptation à la congestion due au démarrage lent, permettant ainsi des transferts de données plus rapides.

Connexions persistantes ou parallèles

Comme nous l'avons vu, les connexions parallèles peuvent accélérer le transfert de pages composites. Cependant, elles présentent certains inconvénients :

• Chaque transaction ouvre/ferme une nouvelle connexion, ce qui coûte du temps et de la bande passante.

• Chaque nouvelle connexion a des performances réduites en raison du démarrage lent du TCP.

• Il existe une limite pratique au nombre de connexions parallèles ouvertes.

Les connexions persistantes offrent certains avantages par rapport aux connexions parallèles. Elles réduisent le délai et la surcharge liés à l'établissement des connexions, maintiennent les connexions à un état optimal et réduisent le nombre potentiel de connexions ouvertes. Cependant, elles doivent être gérées avec soin, sous peine d'accumuler un grand nombre de connexions inactives, consommant ainsi des ressources locales et des ressources sur les clients et serveurs distants.

Les connexions persistantes peuvent être plus efficaces lorsqu'elles sont utilisées conjointement avec des connexions parallèles. Aujourd'hui, de nombreuses applications web ouvrent un petit nombre de connexions parallèles, chacune persistante. Il existe deux types de connexions persistantes : les anciennes connexions HTTP/1.0+ « keep-alive » et les connexions HTTP/1.1 modernes « persistantes ». Nous examinerons ces deux types de connexions dans les sections suivantes.

Connexions Keep-Alive HTTP/1.0+

De nombreux navigateurs et serveurs HTTP/1.0 ont été étendus (à partir de 1996 environ) pour prendre en charge un type expérimental de connexions persistantes, les connexions keep-alive. Ces premières connexions persistantes souffraient de problèmes d'interopérabilité, corrigés dans les versions ultérieures de HTTP/1.1, mais de nombreux clients et serveurs utilisent encore ces anciennes connexions keep-alive.

Certains des avantages en termes de performances des connexions persistantes sont illustrés dans la figure 4-13, qui compare la chronologie de quatre transactions HTTP sur des connexions série à celle des mêmes transactions sur une seule connexion persistante. La chronologie est compressée car les frais de connexion et de fermeture sont supprimés.*

* De plus, les délais de requête et de réponse pourraient également être réduits grâce à la suppression de la phase de démarrage lent. Cet avantage en termes de performances n'est pas illustré dans la figure.

Figure 4-13. Quatre transactions (série et persistante)

Opération Keep-Alive

Keep-alive est obsolète et n'est plus documenté dans la spécification HTTP/1.1 actuelle. Cependant, le protocole de connexion Keep-alive est encore relativement courant dans les navigateurs et les serveurs ; les implémenteurs HTTP doivent donc se préparer à l'utiliser. Nous allons maintenant examiner rapidement le fonctionnement du protocole Keep-alive. Consultez les anciennes versions de la spécification HTTP/1.1 (comme la RFC 2068) pour une explication plus complète du protocole Keep-alive.

Les clients implémentant des connexions Keep-Alive HTTP/1.0 peuvent demander qu'une connexion soit maintenue ouverte en incluant l'en-tête de requête Connection: Keep-Alive.

Si le serveur souhaite maintenir la connexion ouverte pour la requête suivante, il utilisera le même en-tête dans sa réponse (voir Figure 4-14). En l'absence d'en-tête « Connection: keep-alive » dans la réponse, le

client suppose que le serveur ne prend pas en charge le keep-alive et qu'il fermera la connexion lors du retour du message de réponse.

Options de maintien en vie

Notez que les en-têtes de maintien de connexion ne sont que des requêtes visant à maintenir la connexion active. Les clients et les serveurs n'ont pas besoin d'accepter une session de maintien de connexion si elle est demandée. Ils

Figure 4-14. Poignée de main d'en-tête de transaction HTTP/1.0 keep-alive

peuvent fermer les connexions keep-alive inactives à tout moment et sont libres de limiter le nombre de transactions traitées sur une connexion keep-alive.

Le comportement de maintien en activité peut être réglé par des options séparées par des virgules spécifiées dans l'en-tête général Keep-Alive :

• Le paramètre de délai d'expiration est envoyé dans un en-tête de réponse Keep-Alive. Il estime la durée pendant laquelle le serveur est susceptible de maintenir la connexion active. Ceci n'est pas une garantie.

• Le paramètre max est envoyé dans un en-tête de réponse Keep-Alive. Il estime le nombre de transactions HTTP supplémentaires pendant lesquelles le serveur est susceptible de maintenir la connexion active. Ceci n'est pas une garantie.

• L'en-tête Keep-Alive prend également en charge des attributs arbitraires non traités, principalement à des fins de diagnostic et de débogage. La syntaxe est : nom [= valeur].

L'en-tête Keep-Alive est totalement facultatif, mais n'est autorisé que si Connection: Keep-Alive est également présent. Voici un exemple d'en-tête de réponse Keep-Alive indiquant que le serveur prévoit de maintenir la connexion ouverte pendant cinq transactions supplémentaires maximum, ou jusqu'à deux minutes d'inactivité :

Connexion : Keep-Alive

Keep-Alive : max = 5, délai d'attente = 120

Restrictions et règles de connexion Keep-Alive

Voici quelques restrictions et clarifications concernant l'utilisation des connexions keep-alive :

• Le maintien en activité n'est pas activé par défaut dans HTTP/1.0. Le client doit envoyer un

Connexion : en-tête de requête Keep-Alive pour activer les connexions Keep-Alive.

• L'en-tête « Connection : Keep-Alive » doit être envoyé avec tous les messages souhaitant poursuivre la persistance. Si le client n'envoie pas d'en-tête « Connection : Keep-Alive », le serveur fermera la connexion après cette requête.

• Les clients peuvent savoir si le serveur fermera la connexion après la réponse en détectant l'absence de l'en-tête de réponse Connection: Keep-Alive.

• La connexion ne peut être maintenue ouverte que si la longueur du corps de l'entité du message peut être déterminée sans détection de fermeture de connexion. Cela signifie que le corps de l'entité doit avoir une longueur de contenu correcte, un type de média multipartite ou être encodé avec l'encodage de transfert fragmenté. Renvoyer une longueur de contenu incorrecte sur un canal de maintien de connexion est une erreur, car l'autre extrémité de la transaction ne pourra pas détecter précisément la fin d'un message et le début d'un autre.

• Les proxys et les passerelles doivent appliquer les règles de l'en-tête de connexion ; le proxy ou la passerelle doit supprimer tous les champs d'en-tête nommés dans l'en-tête de connexion, ainsi que l'en-tête de connexion lui-même, avant de transférer ou de mettre en cache le message.

• Officiellement, les connexions persistantes ne doivent pas être établies avec un serveur proxy dont la prise en charge de l'en-tête Connection n'est pas garantie, afin d'éviter le problème des proxys passifs décrit ci-dessous. En pratique, cela n'est pas toujours possible.

• Techniquement, tout champ d'en-tête de connexion (y compris « Connection : Keep-Alive ») reçu d'un périphérique HTTP/1.0 doit être ignoré, car il peut avoir été transmis par erreur par un ancien serveur proxy. En pratique, certains clients et serveurs contournent cette règle, même s'ils risquent de se bloquer sur des proxys plus anciens.

• Les clients doivent être prêts à réessayer les demandes si la connexion se ferme avant de recevoir la réponse complète, à moins que la demande puisse avoir des effets secondaires si elle est répétée.

Proxies Keep-Alive et Dumb

Examinons de plus près le problème subtil des proxys keep-alive et dumb. L'en-tête Connection: Keep-Alive d'un client web est censé affecter uniquement le lien TCP quittant le client. C'est pourquoi on l'appelle en-tête « connexion ». Si le client communique avec un serveur web, il envoie un en-tête Connection: Keep-Alive pour indiquer au serveur qu'il souhaite le keep-alive. Le serveur renvoie un en-tête Connection: Keep-Alive s'il prend en charge le keep-alive et ne l'envoie pas dans le cas contraire.

L'en-tête de connexion et les relais aveugles

Le problème vient des proxys, en particulier ceux qui ne comprennent pas l'en-tête de connexion et ignorent qu'ils doivent le supprimer avant de le rediriger vers la chaîne. De nombreux proxys anciens ou simples agissent comme des relais aveugles, acheminant des octets d'une connexion à une autre sans traiter spécifiquement l'en-tête de connexion.

Imaginez un client web communiquant avec un serveur web via un proxy passif agissant comme un relais aveugle. Cette situation est illustrée à la figure 4-15.

Voici ce qui se passe dans cette figure :

1. Dans la figure 4-15a, un client web envoie un message au proxy, incluant l'en-tête « Connexion : Keep-Alive », demandant une connexion persistante si possible. Le client attend une réponse pour savoir si sa demande de canal persistant a été acceptée.

Figure 4-15. Keep-alive n'interagit pas avec les proxys qui ne prennent pas en charge les en-têtes de connexion.

2. Le proxy passif reçoit la requête HTTP, mais ne comprend pas l'en-tête de connexion (il le traite simplement comme un en-tête d'extension). Le proxy ignore ce qu'est le keep-alive et transmet donc le message mot pour mot au serveur (Figure 4-15b). Or, l'en-tête de connexion est un en-tête saut par saut ; il ne s'applique qu'à un seul lien de transport et ne doit pas être transmis au serveur. De mauvaises choses sont sur le point de se produire.

3. Dans la figure 4-15b, la requête HTTP relayée arrive au serveur web. Lorsque le serveur web reçoit l'en-tête proxy « Connection: Keep-Alive », il conclut à tort que le proxy (qui ressemble à n'importe quel autre client pour le serveur) souhaite utiliser le protocole « keep-alive » ! Le serveur web n'y voit aucun inconvénient : il accepte de communiquer avec le proxy et renvoie un en-tête de réponse « Connection: Keep-Alive » (voir la figure 4-15c). À ce stade, le serveur web pense communiquer avec le proxy en mode « keep-alive » et s'en conformera aux règles. Mais le proxy ignore tout du protocole « keep-alive ».

4. Dans la figure 4-15d, le proxy passif relaie le message de réponse du serveur web au client, en lui transmettant l'en-tête « Connexion : Keep-Alive ». Le client voit cet en-tête et suppose que le proxy a accepté de communiquer avec le serveur. À ce stade, le client et le serveur pensent communiquer avec le serveur, mais le proxy avec lequel ils communiquent n'en a aucune idée.

5. Comme le proxy ignore tout du maintien de la connexion active, il renvoie toutes les données reçues au client et attend que le serveur d'origine ferme la connexion. Or, le serveur d'origine ne ferme pas la connexion, pensant que le proxy lui a explicitement demandé de la maintenir ouverte. Le proxy reste donc bloqué en attendant la fermeture de la connexion.

6. Lorsque le client reçoit le message de réponse de la figure 4-15d, il passe directement à la requête suivante et envoie une autre requête au proxy via la connexion persistante (voir figure 4-15e). Le proxy n'attend jamais de nouvelle requête.

sur la même connexion, la requête est ignorée et le navigateur tourne simplement, sans faire de progrès.

7. Cette mauvaise communication provoque le blocage du navigateur jusqu'à ce que le client ou le serveur expire la connexion et la ferme.*

Proxies et en-têtes hop-by-hop

Pour éviter ce type de problème de communication, les proxys modernes ne doivent jamais proxyer l'en-tête Connection ni aucun en-tête dont le nom apparaît dans les valeurs Connection. Ainsi, si un proxy reçoit un en-tête Connection: Keep-Alive, il ne doit proxyer ni l'en-tête Connection ni aucun en-tête nommé Keep-Alive.

De plus, certains en-têtes saut par saut peuvent ne pas être répertoriés comme valeurs d'un en-tête de connexion, mais ne doivent pas non plus être proxyés ni utilisés comme réponse de cache. Il s'agit notamment de Proxy-Authenticate, Proxy-Connection, Transfer-Encoding et Upgrade. Pour plus d'informations, reportez-vous à « L'en-tête de connexion souvent mal compris ».

Le piratage de connexion proxy

Les développeurs de navigateurs et de proxys de Netscape ont proposé une solution astucieuse au problème du relais aveugle, qui ne nécessitait pas la prise en charge des versions avancées de HTTP par toutes les applications web. Cette solution introduisait un nouvel en-tête appelé Proxy-Connection et résolvait le problème d'un relais aveugle unique interposé directement après le client, mais pas dans tous les autres cas. Proxy-Connection est implémenté par les navigateurs modernes lorsque les proxys sont explicitement configurés et est compris par de nombreux proxys.

L'idée est que les proxys passifs se retrouvent en difficulté car ils transmettent aveuglément des en-têtes « hop-by-hop » tels que « Connexion : Keep-Alive ». Ces en-têtes ne sont pertinents que pour une connexion spécifique et ne doivent pas être transmis. Cela pose problème lorsque les en-têtes transmis sont interprétés à tort par les serveurs en aval comme des requêtes du proxy lui-même visant à contrôler sa connexion.

Dans la solution de contournement de Netscape, les navigateurs envoient des en-têtes d'extension Proxy-Connection non standard aux proxys, au lieu des en-têtes Connection officiellement pris en charge et bien connus. Si le proxy est un relais aveugle, il transmet l'en-tête Proxy-Connection non pertinent au serveur web, qui l'ignore sans problème. En revanche, s'il est un proxy intelligent (capable de comprendre les protocoles de connexion persistants), il remplace l'en-tête Proxy-Connection non pertinent par un en-tête Connection, qui est ensuite envoyé au serveur, obtenant ainsi l'effet escompté.

La figure 4-16a–d montre comment un relais aveugle transmet sans danger les en-têtes Proxy-Connection au serveur Web, qui ignore l'en-tête, ce qui empêche toute connexion persistante

* Il existe de nombreux scénarios similaires dans lesquels des échecs se produisent en raison de relais aveugles et de poignées de main transférées.

être établie entre le client et le proxy ou entre le proxy et le serveur. Le proxy intelligent de la figure 4-16e–h interprète l'en-tête Proxy-Connection comme une demande de connexion active et envoie ses propres en-têtes « Connexion : Keep-Alive » pour établir des connexions actives.

Figure 4-16. L'en-tête Proxy-Connection corrige le relais simple aveugle

Ce schéma fonctionne dans les situations où il n'y a qu'un seul proxy entre le client et le serveur. Cependant, s'il y a un proxy intelligent de chaque côté du proxy passif, le problème réapparaît, comme illustré à la figure 4-17.

De plus, il est de plus en plus courant d'observer l'apparition de proxys « invisibles » sur les réseaux, sous forme de pare-feu, de caches d'interception ou d'accélérateurs de serveurs proxy inverses. Ces dispositifs étant invisibles pour le navigateur, celui-ci ne leur envoie pas d'en-têtes Proxy-Connection. Il est essentiel que les applications web transparentes implémentent correctement les connexions persistantes.

Connexions persistantes HTTP/1.1

HTTP/1.1 a progressivement abandonné la prise en charge des connexions persistantes, les remplaçant par une conception améliorée appelée connexions persistantes. Les objectifs des connexions persistantes sont les mêmes que ceux des connexions persistantes, mais leurs mécanismes fonctionnent mieux.

Figure 4-17. La connexion proxy échoue toujours pour les hiérarchies de proxys plus profondes.

Contrairement aux connexions persistantes HTTP/1.0+, les connexions persistantes HTTP/1.1 sont actives par défaut. HTTP/1.1 suppose que toutes les connexions sont persistantes, sauf indication contraire. Les applications HTTP/1.1 doivent explicitement ajouter un en-tête Connection: close à un message pour indiquer qu'une connexion doit être fermée une fois la transaction terminée. Il s'agit d'une différence significative par rapport aux versions précédentes du protocole HTTP, où les connexions persistantes étaient facultatives ou totalement non prises en charge.

Un client HTTP/1.1 suppose qu'une connexion HTTP/1.1 restera ouverte après une réponse, sauf si celle-ci contient un en-tête Connection: close. Cependant, les clients et les serveurs peuvent fermer les connexions inactives à tout moment. Ne pas envoyer Connection: close ne signifie pas que le serveur promet de maintenir la connexion ouverte indéfiniment.

Restrictions et règles de connexion persistante

Voici les restrictions et clarifications concernant l’utilisation des connexions persistantes :

• Après l'envoi d'un en-tête de demande de fermeture de connexion, le client ne peut plus envoyer de demandes sur cette connexion.

• Si un client ne souhaite pas envoyer une autre demande sur la connexion, il doit envoyer un en-tête de demande Connection: close dans la demande finale.

• La connexion peut être maintenue persistante uniquement si tous les messages sur la connexion ont une longueur de message correcte et auto-définie, c'est-à-dire que les corps d'entité doivent avoir des longueurs de contenu correctes ou être codés avec l'encodage de transfert fragmenté.

• Les proxys HTTP/1.1 doivent gérer les connexions persistantes séparément avec les clients et les serveurs : chaque connexion persistante s'applique à un seul saut de transport.

• Les serveurs proxy HTTP/1.1 ne doivent pas établir de connexions persistantes avec un client HTTP/1.0 (en raison des problèmes des anciens proxys qui transmettent les en-têtes de connexion) à moins qu'ils ne connaissent les capacités du client.

En pratique, c’est difficile et de nombreux vendeurs contournent cette règle.

• Quelles que soient les valeurs des en-têtes de connexion, les périphériques HTTP/1.1 peuvent fermer la connexion à tout moment, bien que les serveurs doivent essayer de ne pas fermer au milieu de la

transmission d'un message et doivent toujours répondre à au moins une requête avant de fermer.

• Les applications HTTP/1.1 doivent pouvoir récupérer après une fermeture asynchrone. Les clients doivent réessayer les requêtes tant qu'elles ne présentent pas d'effets secondaires susceptibles de s'accumuler.

• Les clients doivent être prêts à réessayer les demandes si la connexion se ferme avant de recevoir la réponse complète, à moins que la demande puisse avoir des effets secondaires si elle est répétée.

• Un client mono-utilisateur doit maintenir au maximum deux connexions persistantes à un serveur ou proxy, afin d'éviter toute surcharge du serveur. Les proxys pouvant nécessiter davantage de connexions à un serveur pour gérer des utilisateurs simultanés, un proxy doit maintenir au maximum 2N connexions à un serveur ou proxy parent, si N utilisateurs tentent d'accéder aux serveurs.

Connexions par pipeline

HTTP/1.1 permet le pipelining optionnel des requêtes sur les connexions persistantes. Il s'agit d'une optimisation supplémentaire des performances par rapport aux connexions persistantes. Plusieurs requêtes peuvent être mises en file d'attente avant l'arrivée des réponses. Pendant que la première requête transite par le réseau vers un serveur situé à l'autre bout du monde, les deuxième et troisième requêtes peuvent être traitées. Cela peut améliorer les performances dans des conditions de réseau à forte latence, en réduisant les allers-retours.

La figure 4-18a-c montre comment les connexions persistantes peuvent éliminer les retards de connexion TCP et comment les requêtes en pipeline (figure 4-18c) peuvent éliminer les latences de transfert.

Il existe plusieurs restrictions pour le pipeline :

• Les clients HTTP ne doivent pas effectuer de pipeline tant qu’ils ne sont pas sûrs que la connexion est persistante.

• Les réponses HTTP doivent être renvoyées dans le même ordre que les requêtes. Les messages HTTP ne sont pas numérotés ; il est donc impossible de faire correspondre les réponses aux requêtes si elles sont reçues dans le désordre.

• Les clients HTTP doivent être prêts à ce que la connexion se ferme à tout moment et à relancer toute requête en pipeline inachevée. Si le client ouvre une connexion persistante et émet immédiatement 10 requêtes, le serveur est libre de fermer la connexion après avoir traité, par exemple, seulement 5 requêtes. Les 5 requêtes restantes

Connexions par pipeline

Figure 4-18. Quatre transactions (connexions pipelinées)

les demandes échoueront et le client doit être prêt à gérer ces fermetures prématurées et à réémettre les demandes.

• Les clients HTTP ne doivent pas canaliser les requêtes ayant des effets secondaires (comme les POST). En général, en cas d'erreur, le pipeline empêche les clients de savoir lesquelles des requêtes canalisées ont été exécutées par le serveur. Étant donné que les requêtes non idempotentes, comme les POST, ne peuvent pas être relancées en toute sécurité, vous courez le risque que certaines méthodes ne soient jamais exécutées en cas d'erreur.

Les mystères de la connexion proche

La gestion des connexions, et notamment savoir quand et comment les fermer, est l'un des secrets du protocole HTTP. Ce problème est plus subtil que beaucoup de développeurs ne le pensent, et peu d'études ont été menées sur le sujet.

Déconnexion « à volonté »

Tout client, serveur ou proxy HTTP peut fermer une connexion de transport TCP à tout moment. Les connexions sont généralement fermées à la fin d'un message*, mais en cas d'erreur, la connexion peut être fermée au milieu d'une ligne d'en-tête ou à d'autres endroits inhabituels.

Cette situation est fréquente avec les connexions persistantes en pipeline. Les applications HTTP sont libres de fermer les connexions persistantes après un certain temps. Par exemple, après une inactivité prolongée d'une connexion persistante, un serveur peut décider de la fermer.

Cependant, le serveur ne peut jamais être sûr que le client à l'autre bout de la ligne n'était pas sur le point d'envoyer des données au moment même où la connexion « inactive » était fermée par le serveur. Dans ce cas, le client reçoit une erreur de connexion au milieu de l'écriture de son message de requête.

Contenu - Longueur et troncature

Chaque réponse HTTP doit comporter un en-tête Content-Length précis pour décrire la taille du corps de la réponse. Certains serveurs HTTP plus anciens omettent cet en-tête ou incluent une longueur erronée, selon la proximité d'une connexion serveur, pour indiquer la fin réelle des données.

Lorsqu'un client ou un proxy reçoit une réponse HTTP se terminant par une fermeture de connexion et que la longueur réelle de l'entité transférée ne correspond pas à la longueur du contenu (ou qu'il n'y a

pas de longueur du contenu), le récepteur doit remettre en question l'exactitude de la longueur.

Si le récepteur est un proxy de mise en cache, il ne doit pas mettre la réponse en cache (afin de minimiser la multiplication d'une erreur potentielle). Le proxy doit transmettre le message douteux intact, sans tenter de « corriger » la longueur du contenu, afin de préserver la transparence sémantique.

Tolérance de fermeture de connexion, nouvelles tentatives et idempotence

Les connexions peuvent se fermer à tout moment, même en l'absence d'erreur. Les applications HTTP doivent être prêtes à gérer correctement les fermetures inattendues. Si une connexion de transport se ferme pendant que le client effectue une transaction, ce dernier doit la rouvrir.

* Les serveurs ne doivent pas fermer une connexion au milieu d'une réponse, sauf si une défaillance du client ou du réseau est suspectée.

Les mystères de la connexion proche

Connexion et réessayez une fois, sauf si la transaction entraîne des effets secondaires. La situation est pire pour les connexions en pipeline. Le client peut mettre en file d'attente un grand nombre de requêtes, mais le serveur d'origine peut fermer la connexion, laissant de nombreuses requêtes non traitées et nécessitant une reprogrammation.

Les effets secondaires sont importants. Lorsqu'une connexion se ferme après l'envoi de données de requête, mais avant le retour de la réponse, le client ne peut pas être sûr à 100 % de la part de la transaction réellement exécutée par le serveur. Certaines transactions, comme l'obtention d'une page HTML statique, peuvent être répétées indéfiniment sans modification. D'autres, comme l'envoi d'une commande à une librairie en ligne, ne doivent pas être répétées, sous peine de générer plusieurs commandes.

Une transaction est idempotente si elle produit le même résultat, qu'elle soit exécutée une ou plusieurs fois. Les implémenteurs peuvent supposer que les méthodes GET, HEAD, PUT, DELETE, TRACE et OPTIONS partagent cette propriété.* Les clients ne doivent pas canaliser les requêtes non idempotentes (telles que les POST). Dans le cas contraire, une interruption prématurée de la connexion de transport pourrait entraîner des résultats indéterminés. Pour envoyer une requête non idempotente, il est conseillé d'attendre le statut de réponse de la requête précédente.

Les méthodes ou séquences non idempotentes ne doivent pas être réexécutées automatiquement, bien que les agents utilisateurs puissent offrir à un opérateur humain la possibilité de réexécuter la

requête. Par exemple, la plupart des navigateurs proposent une boîte de dialogue lors du rechargement d'une réponse POST en cache, vous demandant si vous souhaitez réexécuter la transaction.

Fermeture de la connexion gracieuse

Les connexions TCP sont bidirectionnelles, comme illustré à la figure 4-19. Chaque côté d'une connexion TCP possède une file d'attente d'entrée et une file d'attente de sortie, pour les données lues ou écrites. Les données placées en sortie d'un côté apparaîtront à l'entrée de l'autre côté.

Figure 4-19. Les connexions TCP sont bidirectionnelles

Fermetures complètes et à moitié fermées

Une application peut fermer l'un ou les deux canaux d'entrée et de sortie TCP. Un appel de socket close() ferme les canaux d'entrée et de sortie d'une connexion TCP.

*Les administrateurs qui utilisent des formulaires dynamiques basés sur GET doivent s'assurer que les formulaires sont idempotents.

Ceci est appelé « fermeture complète » et est illustré à la figure 4-20a. Vous pouvez utiliser l'appel de socket shutdown() pour fermer individuellement le canal d'entrée ou de sortie. Ceci est appelé « fermeture partielle » et est illustré à la figure 4-20b.

Figure 4-20. Fermeture complète et à moitié fermée

Erreurs de fermeture et de réinitialisation TCP

Les applications HTTP simples ne peuvent utiliser que des fermetures complètes. Cependant, lorsqu'elles commencent à communiquer avec de nombreux autres types de clients, serveurs et proxys HTTP, et qu'elles commencent à utiliser des connexions persistantes en pipeline, il devient important d'utiliser des demi-fermetures pour éviter que leurs homologues ne subissent des erreurs d'écriture inattendues.

En général, la fermeture du canal de sortie de votre connexion est toujours sûre. Le pair de l'autre côté de la connexion sera averti de la fermeture de la connexion par une notification de fin de flux une fois toutes les données lues depuis son tampon.

Fermer le canal d'entrée de votre connexion est plus risqué, sauf si vous savez que l'autre côté n'envisage plus d'envoyer de données. Si l'autre côté envoie des données sur votre canal d'entrée fermé, le système d'exploitation renvoie un message TCP « connexion réinitialisée par l'homologue » à la machine de l'autre côté, comme illustré à la figure 4-21. La plupart des systèmes d'exploitation

considèrent cela comme une erreur grave et effacent toutes les données en mémoire tampon que l'autre côté n'a pas encore lues. Ceci est très néfaste pour les connexions pipelinées.

Imaginons que vous ayez envoyé 10 requêtes en pipeline sur une connexion persistante, et que les réponses soient déjà arrivées et se trouvent dans la mémoire tampon de votre système d'exploitation (mais que l'application ne les ait pas encore lues). Imaginons maintenant que vous envoyiez la requête n° 11, mais que le serveur décide que vous avez utilisé cette connexion suffisamment longtemps et la ferme. Votre requête n° 11 arrivera sur une connexion fermée et vous enverra une réinitialisation. Cette réinitialisation effacera vos mémoires tampons d'entrée.

Les mystères de la connexion proche

Figure 4-21. Les données arrivant à une connexion fermée génèrent l'erreur « Réinitialisation de la connexion par l'homologue ».

Lorsque vous parvenez enfin à lire les données, vous obtiendrez une erreur de réinitialisation de connexion par l'homologue et les données de réponse mises en mémoire tampon et non lues seront perdues, même si une grande partie d'entre elles sont arrivées avec succès sur votre machine.

Fermeture gracieuse

La spécification HTTP conseille que lorsque les clients ou les serveurs souhaitent fermer une connexion de manière inattendue, ils doivent « émettre une fermeture gracieuse sur la connexion de transport », mais elle ne décrit pas comment procéder.

En général, les applications implémentant des fermetures progressives ferment d'abord leurs canaux de sortie, puis attendent que le pair de l'autre côté de la connexion ferme les siens. Une fois que les deux côtés ont fini de s'indiquer qu'ils n'enverront plus de données (c'est-à-dire en fermant les canaux de sortie), la connexion peut être complètement fermée, sans risque de réinitialisation.

Malheureusement, rien ne garantit que le pair implémente ou vérifie les demi-fermetures. C'est pourquoi les applications souhaitant se fermer correctement doivent fermer à moitié leurs canaux de sortie et vérifier périodiquement l'état de leurs canaux d'entrée (recherche de données ou de fin de flux). Si le canal d'entrée n'est pas fermé par le pair dans un délai donné, l'application peut forcer la fermeture de la connexion pour économiser des ressources.

Pour plus d'informations

Ceci conclut notre aperçu du secteur de la plomberie HTTP. Veuillez consulter les sources de référence suivantes pour plus d'informations

sur les performances TCP et les fonctionnalités de gestion des connexions HTTP.

Connexions HTTP

http://www.ietf.org/rfc/rfc2616.txt

La RFC 2616, « Hypertext Transfer Protocol—HTTP/1.1 », est la spécification officielle de HTTP/1.1 ; elle explique l'utilisation des champs d'en-tête HTTP pour la mise en œuvre de connexions HTTP parallèles, persistantes et pipelinées. Ce document ne traite pas de l'utilisation correcte des connexions TCP sous-jacentes.

http://www.ietf.org/rfc/rfc2068.txt

La RFC 2068 est la version 1997 du protocole HTTP/1.1. Elle contient une explication des connexions Keep-Alive HTTP/1.0+, absente de la RFC 2616.

http://www.ics.uci.edu/pub/ietf/http/draft-ietf-http-connection-00.txt

Ce projet Internet expiré, « Gestion des connexions HTTP », contient une bonne discussion sur les problèmes auxquels est confrontée la gestion des connexions HTTP.

Problèmes de performances HTTP

http://www.w3.org/Protocols/HTTP/Performance/

Cette page Web du W3C, intitulée « Présentation des performances HTTP », contient quelques articles et outils liés aux performances HTTP et à la gestion des connexions.

http://www.w3.org/Protocols/HTTP/1.0/HTTPPerformance.html

Ce court mémo de Simon Spero, « Analyse des problèmes de performances HTTP », est l'une des premières (1994) évaluations des performances des connexions HTTP. Il fournit quelques premières mesures de performance concernant l'impact de l'établissement de la connexion, de la lenteur du démarrage et de l'absence de partage de connexion.

ftp://gatekeeper.dec.com/pub/DEC/WRL/research-reports/WRL-TR-95.4.pdf

« Priorité au protocole HTTP à connexion persistante ». http://www.isi.edu/lsam/publications/phttp_tcp_interactions/paper.html

« Interactions de performances entre les implémentations P-HTTP et TCP. » http://www.sun.com/sun-on-net/performance/tcp.slowstart.html

« Réglage du démarrage lent TCP pour Solaris » est une page web de Sun Microsystems qui aborde certaines implications pratiques du

démarrage lent TCP. C'est une lecture utile, même si vous travaillez avec différents systèmes d'exploitation.

TCP/IP

Les trois ouvrages suivants de W. Richard Stevens sont d'excellents ouvrages d'ingénierie détaillés sur TCP/IP. Ils sont extrêmement utiles à tous les utilisateurs de TCP :

TCP Illustrated, Volume I : Les protocoles W. Richard Stevens, Addison Wesley

Programmation réseau UNIX, volume 1 : API réseau

W. Richard Stevens, Prentice-Hall

Programmation réseau UNIX, volume 2 : la mise en œuvre

W. Richard Stevens, Prentice-Hall

Pour plus d'informations

Les documents et spécifications suivants décrivent TCP/IP et les fonctionnalités qui affectent ses performances. Certaines de ces spécifications datent de plus de 20 ans et, compte tenu du succès mondial de TCP/IP, peuvent probablement être considérées comme des trésors historiques :

Dans « Rethinking the TCP Nagle Algorithm », Jeff Mogul et Greg Minshall présentent une perspective moderne sur l'algorithme de Nagle, décrivent les applications qui devraient et ne devraient pas utiliser l'algorithme et proposent plusieurs modifications.

http://www.ietf.org/rfc/rfc2001.txt

La RFC 2001, « Algorithmes de démarrage lent TCP, d'évitement de congestion, de retransmission rapide et de récupération rapide », définit l'algorithme de démarrage lent TCP.

http://www.ietf.org/rfc/rfc1122.txt

La RFC 1122, « Exigences pour les hôtes Internet – Couches de communication », traite de l’accusé de réception TCP et des accusés de réception différés.

http://www.ietf.org/rfc/rfc896.txt

La RFC 896, « Contrôle de congestion dans les réseaux IP/TCP », a été publiée par John Nagle en 1984. Elle décrit la nécessité du contrôle de congestion TCP et introduit ce que l'on appelle désormais « l'algorithme de Nagle ».

http://www.ietf.org/rfc/rfc0813.txt

La RFC 813, « Stratégie de fenêtre et d'accusé de réception dans TCP », est une spécification historique (1982) qui décrit les stratégies

d'implémentation de fenêtre et d'accusé de réception TCP et fournit une description précoce de la technique d'accusé de réception différé.

http://www.ietf.org/rfc/rfc0793.txt

RFC 793, « Transmission Control Protocol », est la définition classique du protocole TCP donnée par Jon Postel en 1981.

PARTIE II

II. Architecture HTTP

Les six chapitres de la deuxième partie mettent en évidence les applications serveur HTTP, proxy, cache, passerelle et robot, qui sont les éléments constitutifs de l'architecture des systèmes Web :

• Le chapitre 5, Serveurs Web, donne un aperçu des architectures de serveurs Web.

• Le chapitre 6, Proxies, décrit les serveurs proxy HTTP, qui sont des serveurs intermédiaires qui connectent les clients HTTP et agissent comme des plates-formes pour les services et les contrôles HTTP.

• Le chapitre 7, Mise en cache, se penche sur la science des caches Web, des dispositifs qui améliorent les performances et réduisent le trafic en créant des copies locales de documents populaires.

• Le chapitre 8, Points d'intégration : passerelles, tunnels et relais, explique les applications qui permettent à HTTP d'interagir avec des logiciels qui parlent différents protocoles, y compris les protocoles cryptés SSL.

• Le chapitre 9, Web Robots, conclut notre visite de l'architecture HTTP avec les clients Web. • Le chapitre 10, HTTP-NG, couvre les sujets futurs pour HTTP, en particulier HTTP-NG.

www.it-ebooks.info

Chapitre 5Ceci est le titre du livre CHAPITRE 5

Serveurs Web

Les serveurs web génèrent des milliards de pages web chaque jour. Ils vous donnent la météo, chargent vos paniers d'achat en ligne et vous permettent de retrouver vos anciens camarades de lycée. Les serveurs web sont les bêtes de somme du World Wide Web. Dans ce chapitre, nous :

• Examiner les nombreux types différents de serveurs Web logiciels et matériels.

• Décrivez comment écrire un serveur Web de diagnostic simple en Perl.

• Expliquez comment les serveurs Web traitent les transactions HTTP, étape par étape.

Lorsque cela permet de concrétiser les choses, nos exemples utilisent le serveur Web Apache et ses options de configuration.

Les serveurs Web sont de toutes formes et de toutes tailles

Un serveur web traite les requêtes HTTP et fournit les réponses. Le terme « serveur web » peut désigner soit un logiciel serveur, soit l'appareil ou l'ordinateur dédié à la gestion des pages web.

Il existe des serveurs web de toutes sortes, de toutes formes et de toutes tailles. On trouve des serveurs web triviaux avec des scripts Perl de 10 lignes, des moteurs de commerce sécurisés de 50 Mo et de minuscules serveurs sur une carte. Mais quelles que soient leurs différences fonctionnelles, tous les serveurs web reçoivent des requêtes HTTP pour les ressources et renvoient le contenu aux clients (voir la figure 1-5).

Implémentations de serveurs Web

Les serveurs Web implémentent HTTP et la gestion des connexions TCP associée. Ils gèrent également les ressources servies par le serveur Web et fournissent des fonctionnalités d'administration pour configurer, contrôler et améliorer le serveur Web.

La logique du serveur Web implémente le protocole HTTP, gère les ressources Web et fournit des fonctionnalités d'administration. Elle partage la gestion des connexions TCP avec le système d'exploitation.

Le système d'exploitation gère les détails matériels du système informatique sous-jacent et fournit un support réseau TCP/IP, des systèmes de fichiers pour contenir les ressources Web et une gestion des processus pour contrôler les activités informatiques en cours.

Les serveurs Web sont disponibles sous de nombreuses formes :

• Vous pouvez installer et exécuter des serveurs Web logiciels à usage général sur des systèmes informatiques standard.

• Si vous ne voulez pas vous embêter à installer un logiciel, vous pouvez acheter un serveur Web, dans lequel le logiciel est préinstallé et préconfiguré sur un ordinateur, souvent dans un châssis élégant.

• Étant donné les miracles des microprocesseurs, certaines entreprises proposent même des serveurs Web intégrés implémentés dans un petit nombre de puces informatiques, ce qui en fait des consoles d’administration parfaites pour les appareils grand public.

Examinons chacun de ces types d'implémentations.

Serveurs Web logiciels à usage général

Les serveurs web logiciels à usage général fonctionnent sur des systèmes informatiques standard connectés au réseau. Vous pouvez choisir des logiciels open source (comme Apache ou Jigsaw du W3C) ou commerciaux (comme les serveurs web de Microsoft et d'iPlanet). Les logiciels de serveur web sont disponibles pour la quasi-totalité des ordinateurs et des systèmes d'exploitation.

Bien qu'il existe des dizaines de milliers de types différents de programmes de serveur Web (y compris des serveurs Web personnalisés et à usage spécifique), la plupart des logiciels de serveur Web proviennent d'un petit nombre d'organisations.

En février 2002, l'enquête Netcraft (http://www.netcraft.com/survey/) a montré que trois fournisseurs dominaient le marché des serveurs Web Internet publics (voir Figure 5-1) :

• Le logiciel gratuit Apache alimente près de 60 % de tous les serveurs Web Internet.

• Le serveur Web Microsoft représente 30 % supplémentaires.

• Les serveurs Sun iPlanet représentent 3 % supplémentaires.

Figure 5-1. Part de marché des serveurs Web estimée par l'enquête automatisée de Netcraft

Il faut toutefois prendre ces chiffres avec des pincettes, car l'enquête Netcraft est généralement perçue comme surestimant la domination du logiciel Apache. Premièrement, l'enquête comptabilise les serveurs indépendamment de leur popularité. Les études d'accès aux serveurs proxy menées par les principaux FAI suggèrent que le nombre de pages servies par les serveurs Apache est bien inférieur à 60 %, mais dépasse néanmoins celui de Microsoft et de Sun iPlanet. De plus, il semblerait que les serveurs Microsoft et iPlanet soient plus populaires qu'Apache au sein des entreprises.

Appareils de serveur Web

Les serveurs web sont des solutions logicielles et matérielles pré-installées. Le fournisseur préinstalle un serveur logiciel sur une plateforme informatique choisie par le fournisseur et préconfigure le logiciel. Voici quelques exemples de serveurs web :

• Appareils Web Sun/Cobalt RaQ (http://www.cobalt.com)

• Toshiba Magnia SG10 (http://www.toshiba.com)

• Dispositif de serveur Web IBM Whistle (http://www.whistle.com)

Les solutions matérielles-logicielles suppriment l'installation et la configuration de logiciels et simplifient souvent considérablement l'administration. Cependant, le serveur web est souvent moins flexible et moins riche en fonctionnalités, et le matériel du serveur est difficilement adaptable ou évolutif.

Serveurs Web intégrés

Les serveurs embarqués sont de petits serveurs web destinés à être intégrés à des produits grand public (par exemple, des imprimantes ou des appareils électroménagers). Ils permettent aux utilisateurs d'administrer leurs appareils grand public via une interface de navigateur web conviviale.

Certains serveurs web embarqués peuvent même être déployés sur moins d'un pouce carré, mais ils offrent généralement un ensemble minimal de fonctionnalités. Voici deux exemples de serveurs web embarqués de très petite taille :

• Serveur Web de la taille d'une tête d'allumette IPic (http://www-ccs.cs.umass.edu/~shri/iPic.html)

• Serveur Web Ethernet NetMedia SitePlayer SP1 (http://www.siteplayer.com)

Un serveur Web Perl minimal

Si vous souhaitez créer un serveur HTTP complet, vous avez du travail à faire. Le cœur du serveur web Apache compte plus de 50 000 lignes de code, et les modules de traitement optionnels augmentent considérablement ce nombre.

Tous ces logiciels sont nécessaires pour prendre en charge les fonctionnalités HTTP/1.1 : prise en charge de ressources riches, hébergement virtuel, contrôle d'accès, journalisation, configuration, surveillance et performances. Cela dit, vous pouvez créer un serveur HTTP minimalement fonctionnel en moins de 30 lignes de code Perl. Voyons voir.

Un serveur Web Perl minimal

L'exemple 5-1 présente un petit programme Perl appelé type-o-serve. Ce programme est un outil de diagnostic utile pour tester les interactions avec les clients et les proxys. Comme tout serveur web, type-o-serve attend une connexion HTTP. Dès qu'il reçoit le message de requête, il l'affiche à l'écran ; il attend ensuite que vous saisissiez (ou colliez) un message de réponse, qui est renvoyé au client. De cette façon, type-o-serve se fait passer pour un serveur web, enregistre les messages de requête HTTP exacts et vous permet de renvoyer n'importe quel message de réponse HTTP.

Cet utilitaire type-o-serve simple n'implémente pas la plupart des fonctionnalités HTTP, mais il est utile pour générer des messages de

réponse serveur, de la même manière que Telnet génère des messages de requête client (voir l'exemple 5-1). Vous pouvez télécharger le programme type-o-serve à l'adresse http://www.http-guide.com/tools/type-o-serve.pl.

Exemple 5-1. type-o-serve : un serveur Web Perl minimal utilisé pour le débogage HTTP

```
#!/usr/bin/perl

utiliser Socket; utiliser Carp; utiliser FileHandle;

# (1) utiliser le port 8080 par défaut, sauf s'il est remplacé sur la ligne de commande

$port = (@ARGV ? $ARGV[0] : 8080);

# (2) créer un socket TCP local et le configurer pour écouter les connexions

$proto = getprotobyname('tcp');

socket(S, PF_INET, SOCK_STREAM, $proto) || mourir; setsockopt(S, SOL_SOCKET, SO_REUSEADDR, pack("l", 1)) || mourir; bind(S, sockaddr_in($port, INADDR_ANY)) || mourir; écouter(S, SOMAXCONN) || mourir;

# (3) imprimer un message de démarrage

printf(" <<<Type-O-Serve Accepté sur le port %d>>>\n\n",$port);

tandis que (1)

{

   # (4) attendre une connexion C

   $cport_caddr = accept(C, S);

   ($cport,$caddr) = sockaddr_in($cport_caddr);

   C->autoflush(1);

   # (5) indiquez de qui provient la connexion

   $cname = gethostbyaddr($caddr,AF_INET); printf(" <<<Requête de '%s'>>>\n",$cname);

   # (6) lire le message de demande jusqu'à la ligne vide et l'imprimer à l'écran pendant que ($line = <C>)

   {

      imprimer $line; si ($line =~ /^\r/) { dernier; }

   }
```

Exemple 5-1. type-o-serve : un serveur Web Perl minimal utilisé pour le débogage HTTP (suite)

```
   # (7) invite à saisir un message de réponse et des lignes de réponse,

   # envoi de lignes de réponse au client, jusqu'à solitaire "." printf("
<<<Type Réponse suivie de '.'>>>\n");

   tandis que ($line = <STDIN>)

   {

      $line =~ s/\r//; $line =~ s/\n//; if ($line =~ /^\./) { last; } print C
$line . "\r\n";

   } fermer(C); }
```

La figure 5-2 montre comment l'administrateur de la quincaillerie de Joe peut utiliser type-oserve pour tester la communication HTTP :

• Tout d'abord, l'administrateur démarre le serveur de diagnostic type-o-serve, en écoutant sur un port spécifique. Comme la quincaillerie Joe's possède déjà un serveur web de production sur le port 80, l'administrateur démarre le serveur type-o-serve sur le port 8080 (vous pouvez choisir n'importe quel port inutilisé) avec la ligne de commande suivante :

```
% type-o-serve.pl 8080
```

• Une fois type-o-serve exécuté, vous pouvez pointer un navigateur vers ce serveur Web. Dans la figure 5-2, nous accédons à http://www.joes-hardware.com:8080/foo/bar/blah.txt.

Le programme type-o-serve reçoit la requête HTTP du navigateur et affiche son contenu à l'écran. L'outil de diagnostic type-o-serve attend ensuite que l'utilisateur saisisse une réponse simple, suivie d'un point sur une ligne vide.

• type-o-serve renvoie le message de réponse HTTP au navigateur, et le navigateur affiche le corps du message de réponse.

Ce que font les vrais serveurs Web

Le serveur Perl présenté dans l'exemple 5-1 est un exemple simple de serveur web. Les serveurs web commerciaux les plus récents sont beaucoup plus complexes, mais ils effectuent plusieurs tâches courantes, comme le montre la figure 5-3 :

1. Configurer la connexion : accepter une connexion client ou la fermer si le client n'est pas souhaité.

2. Recevoir une demande : lire un message de demande HTTP à partir du réseau.

3. Traiter la demande : interpréter le message de demande et prendre des mesures.

4. Accéder à la ressource : accédez à la ressource spécifiée dans le message.

5. Construire la réponse : créez le message de réponse HTTP avec les en-têtes appropriés.

6. Envoyer une réponse : renvoyez la réponse au client.

7. Enregistrer la transaction : placez des notes sur la transaction terminée dans un fichier journal.

Ce que font les vrais serveurs Web

Figure 5-2. L'utilitaire type-o-serve permet de saisir les réponses du serveur à renvoyer aux clients.

Figure 5-3. Étapes d'une requête de serveur Web de base

Les sept sections suivantes mettent en évidence la manière dont les serveurs Web exécutent ces tâches de base.

Étape 1 : Accepter les connexions client

Si un client dispose déjà d'une connexion persistante au serveur, il peut l'utiliser pour envoyer sa requête. Sinon, il doit ouvrir une nouvelle connexion au serveur (voir le chapitre 4 pour plus d'informations sur la technologie de gestion des connexions HTTP).

Gestion des nouvelles connexions

Lorsqu'un client demande une connexion TCP au serveur Web, le serveur Web établit la connexion et détermine quel client se trouve de l'autre côté de la connexion, en extrayant l'adresse IP de la connexion TCP.* Une fois qu'une nouvelle connexion est établie et acceptée, le serveur ajoute la nouvelle connexion à sa liste de connexions de serveur Web existantes et se prépare à surveiller les données sur la connexion.

Le serveur web est libre de rejeter et de fermer immédiatement toute connexion. Certains serveurs web ferment les connexions si l'adresse IP ou le nom d'hôte du client est non autorisé ou s'il s'agit d'un client malveillant connu. D'autres techniques d'identification peuvent également être utilisées.

Identification du nom d'hôte du client

La plupart des serveurs web peuvent être configurés pour convertir les adresses IP des clients en noms d'hôtes, grâce au DNS inversé. Les serveurs web peuvent utiliser le nom d'hôte du client pour un contrôle d'accès et une journalisation détaillés. Attention : la recherche de noms d'hôtes peut être très longue, ce qui ralentit les transactions web. De

nombreux serveurs web haute capacité désactivent la résolution des noms d'hôtes ou ne l'activent que pour certains contenus.

Vous pouvez activer la recherche de noms d'hôtes dans Apache avec la directive de configuration HostnameLookups. Par exemple, les directives de configuration Apache de l'exemple 5-2 activent la résolution de noms d'hôtes uniquement pour les ressources HTML et CGI.

Exemple 5-2. Configuration d'Apache pour rechercher les noms d'hôtes des ressources HTML et CGI

Recherche de nom d'hôte désactivée

<Fichiers ~ "\.(html|htm|cgi)$">

Recherches de noms d'hôtes sur

</Fichiers>

Déterminer l'utilisateur client via l'identifiant

Certains serveurs web prennent également en charge le protocole IETF ident. Ce protocole permet aux serveurs de déterminer quel nom d'utilisateur a initié une connexion HTTP. Cette information est

* Les différents systèmes d'exploitation ont des interfaces et des structures de données différentes pour manipuler les connexions TCP. Dans les environnements Unix, la connexion TCP est représentée par un socket, et l'adresse IP du client peut être obtenue à partir de ce socket grâce à l'appel getpeername.

Étape 1 : Accepter les connexions client

particulièrement utile pour la journalisation du serveur Web : le deuxième champ du format de journal commun populaire contient le nom d'utilisateur d'identification de chaque requête HTTP.*

Si un client prend en charge le protocole ident, il écoute les requêtes ident sur le port TCP 113. La figure 5-4 illustre le fonctionnement du protocole ident. Dans la figure 5-4a, le client ouvre une connexion HTTP. Le serveur ouvre ensuite sa propre connexion vers le port identd du client (113), envoie une simple requête demandant le nom d'utilisateur correspondant à la nouvelle connexion (spécifié par les numéros de port client et serveur), et récupère la réponse du client contenant le nom d'utilisateur.

Figure 5-4. Utilisation du protocole ident pour déterminer le nom d'utilisateur du client HTTP

ident peut fonctionner au sein des organisations, mais il ne fonctionne pas bien sur l'Internet public pour de nombreuses raisons, notamment :

• De nombreux PC clients n'exécutent pas le logiciel démon du protocole d'identification identd.

• Le protocole ident retarde considérablement les transactions HTTP.

• De nombreux pare-feu n'autorisent pas le trafic d'identification entrant.

• Le protocole d’identification n’est pas sécurisé et facile à fabriquer.

• Le protocole ident ne prend pas bien en charge les adresses IP virtuelles.

• L’exposition des noms d’utilisateur des clients soulève des préoccupations en matière de confidentialité.

Vous pouvez indiquer aux serveurs web Apache d'utiliser la recherche d'identité avec la directive IdentityCheck d'Apache. Si aucune information d'identité n'est disponible, Apache remplit les champs du journal d'identité avec des tirets (-). Les fichiers journaux au format CLF contiennent généralement des tirets dans le deuxième champ, car aucune information d'identité n'est disponible.

Étape 2 : Réception des messages de demande

Lorsque les données arrivent sur les connexions, le serveur Web lit les données de la connexion réseau et analyse les éléments du message de demande (Figure 5-5).

* Ce champ d'identification du format de journal commun est appelé « rfc931 », d'après une version obsolète de la RFC définissant le protocole d'identification (la spécification d'identification mise à jour est documentée par la RFC 1413).

Figure 5-5. Lecture d'un message de requête depuis une connexion

Lors de l'analyse du message de demande, le serveur Web :

• Analyse la ligne de requête à la recherche de la méthode de requête, de l'identifiant de ressource spécifié (URI) et du numéro de version*, chacun séparé par un seul espace et se terminant par une séquence de retour chariot-saut de ligne (CRLF)†

• Lit les en-têtes de message, chacun se terminant par CRLF

• Détecte la ligne vide de fin d'en-tête, se terminant par CRLF (si présent)

• Lit le corps de la requête, le cas échéant (longueur spécifiée par l'en-tête Content-Length)

Lors de l'analyse des messages de requête, les serveurs web reçoivent des données d'entrée du réseau de manière irrégulière. La connexion réseau peut être interrompue à tout moment. Le serveur web doit lire

les données du réseau et stocker temporairement les données partielles du message en mémoire jusqu'à ce qu'il reçoive suffisamment de données pour les analyser et les interpréter.

Représentations internes des messages

Certains serveurs web stockent également les messages de requête dans des structures de données internes, facilitant ainsi leur manipulation. Par exemple, la structure de données peut contenir les pointeurs et la longueur de chaque partie du message de requête, et les en-têtes peuvent être stockés dans une table de correspondance rapide afin d'accéder rapidement à leurs valeurs spécifiques (Figure 5-6).

Architectures de traitement d'entrée/sortie de connexion

Les serveurs web hautes performances prennent en charge des milliers de connexions simultanées. Ces connexions permettent au serveur web de communiquer avec des clients du monde entier, chacun disposant d'une ou plusieurs connexions ouvertes. Certaines de ces connexions peuvent envoyer des requêtes rapidement au serveur web, tandis que d'autres sont lentes.

* La version initiale de HTTP, appelée HTTP/0.9, ne prend pas en charge les numéros de version. Certains serveurs web prennent en charge les numéros de version manquants, interprétant le message comme une requête HTTP/0.9.

† De nombreux serveurs Web prennent en charge LF ou CRLF comme séquences de fin de ligne, car certains clients envoient par erreur LF comme terminateur de fin de ligne.

Étape 2 : Réception des messages de demande

Figure 5-6. Analyse d'un message de requête pour en faire une représentation interne pratique

les demandes sont lentes ou peu fréquentes, et d’autres encore restent inactives, attendant tranquillement une activité future.

Les serveurs web sont constamment à l'affût des nouvelles requêtes, car elles peuvent arriver à tout moment. Les architectures de serveurs web traitent les requêtes de différentes manières, comme l'illustre la figure 5-7 :

Serveurs Web monothread (Figure 5-7a)

Les serveurs web monothread traitent une requête à la fois jusqu'à son achèvement. Une fois la transaction terminée, la connexion suivante est traitée. Cette architecture est simple à mettre en œuvre, mais pendant le traitement, toutes les autres connexions sont ignorées. Cela engendre de sérieux problèmes de performances et ne convient qu'aux

serveurs à faible charge et aux outils de diagnostic comme type-o-serve.

Serveurs Web multiprocessus et multithread (Figure 5-7b)

Les serveurs Web multiprocessus et multithreads dédient plusieurs processus ou threads plus efficaces pour traiter les requêtes simultanément.* Les threads/processus peuvent être créés à la demande ou à l'avance.† Certains serveurs dédient un thread/processus pour chaque connexion, mais lorsqu'un serveur traite des centaines, des milliers, voire des dizaines de milliers de connexions simultanées, le nombre de processus ou de threads résultant peut consommer trop de mémoire ou de système

* Un processus est un flux de contrôle de programme individuel, doté de son propre ensemble de variables. Un thread est une version plus rapide et plus efficace d'un processus. Tant les threads que les processus permettent à un même programme d'effectuer plusieurs tâches simultanément. Pour simplifier, nous traitons processus et threads de manière interchangeable. Cependant, en raison des différences de performances, de nombreux serveurs hautes performances sont à la fois multiprocessus et multithreads.

† Les systèmes dans lesquels les threads sont créés à l'avance sont appelés systèmes de « pool de travailleurs », car un ensemble de threads attend dans un pool que du travail soit effectué.

ressources. Ainsi, de nombreux serveurs Web multithreads imposent une limite au nombre maximal de threads/processus.

Serveurs d'E/S multiplexés (Figure 5-7c)

Pour prendre en charge un grand nombre de connexions, de nombreux serveurs web adoptent des architectures multiplexées. Dans une architecture multiplexée, l'activité de toutes les connexions est surveillée simultanément. Lorsqu'une connexion change d'état (par exemple, lorsque des données deviennent disponibles ou qu'une erreur survient), un léger traitement est effectué sur la connexion ; une fois ce traitement terminé, la connexion est renvoyée dans la liste des connexions ouvertes pour le prochain changement d'état. Une connexion n'est traitée que lorsqu'une tâche est à effectuer ; les threads et les processus ne sont pas bloqués en attente de connexions inactives.

Serveurs Web multithreads multiplexés (Figure 5-7d)

Certains systèmes combinent le multithreading et le multiplexage pour tirer parti des multiples processeurs de la plateforme informatique. Plusieurs threads (souvent un par processeur physique) surveillent chacun les connexions ouvertes (ou un sous-ensemble de connexions ouvertes) et effectuent une petite quantité de travail sur chaque connexion.

Figure 5-7. Architectures d'entrée/sortie du serveur Web

Étape 2 : Réception des messages de demande

Étape 3 : Traitement des demandes

Une fois que le serveur Web a reçu une demande, il peut traiter la demande à l'aide de la méthode, de la ressource, des en-têtes et du corps facultatif.

Certaines méthodes (par exemple, POST) requièrent des données de corps d'entité dans le message de requête. D'autres méthodes (par exemple, OPTIONS) autorisent un corps de requête, mais n'en exigent pas. Quelques méthodes (par exemple, GET) interdisent les données de corps d'entité dans les messages de requête.

Nous ne parlerons pas ici du traitement des requêtes, car c'est le sujet de la plupart des chapitres du reste de ce livre !

Étape 4 : Cartographie et accès aux ressources

Les serveurs Web sont des serveurs de ressources. Ils fournissent du contenu pré-créé, tel que des pages HTML ou des images JPEG, ainsi que du contenu dynamique provenant d'applications génératrices de ressources exécutées sur les serveurs.

Avant que le serveur Web puisse fournir du contenu au client, il doit identifier la source du contenu, en mappant l'URI du message de demande au contenu approprié ou au générateur de contenu sur le serveur Web.

Docroots

Les serveurs Web prennent en charge différents types de mappage de ressources, mais la forme la plus simple utilise l'URI de la requête pour nommer un fichier dans le système de fichiers du serveur Web. Généralement, un dossier spécial du système de fichiers du serveur Web est réservé au contenu Web. Ce dossier est appelé « racine du document », ou docroot. Le serveur Web extrait l'URI du message de requête et l'ajoute à la racine du document.

Dans la figure 5-8, une requête arrive pour /specials/saw-blade.gif. Dans cet exemple, le serveur web a la racine de document /usr/local/httpd/files. Le serveur web renvoie le fichier /usr/local/httpd/files/specials/saw-blade.gif.

Figure 5-8. Mappage de l'URI de la requête à la ressource du serveur Web local

Pour définir la racine du document pour un serveur Web Apache, ajoutez une ligne DocumentRoot au fichier de configuration httpd.conf :

DocumentRoot /usr/local/httpd/files

Les serveurs veillent à ce que les URL relatives ne soient pas sauvegardées hors d'une racine de documentation et n'exposent pas d'autres parties du système de fichiers. Par exemple, la plupart des serveurs web matures n'autorisent pas cette URI à accéder aux fichiers situés au-dessus de la racine du document Joe's Hardware :

http://www.joes-hardware.com/../

Docroots hébergés virtuellement

Les serveurs web hébergés virtuellement hébergent plusieurs sites web sur le même serveur, attribuant à chaque site sa propre racine de document distincte. Un serveur web hébergé virtuellement identifie la racine de document appropriée à utiliser à partir de l'adresse IP ou du nom d'hôte dans l'URI ou l'en-tête Host. Ainsi, deux sites web hébergés sur le même serveur peuvent avoir un contenu totalement distinct, même si les URI de requête sont identiques.

Dans la figure 5-9, le serveur héberge deux sites : www.joes-hardware.com et www.marysantiques.com. Le serveur peut distinguer les sites web grâce à l'en-tête HTTP Host ou à des adresses IP distinctes.

• Lorsque la requête A arrive, le serveur récupère le fichier /docs/joe/index.html.

• Lorsque la requête B arrive, le serveur récupère le fichier /docs/mary/index.html.

Figure 5-9. Différents docroots pour les requêtes hébergées virtuellement

La configuration de docroots hébergés virtuellement est simple pour la plupart des serveurs web. Pour le serveur web Apache, vous devez configurer un bloc VirtualHost pour chaque site web virtuel et inclure le bloc DocumentRoot pour chaque serveur virtuel (exemple 5-3).

Exemple 5-3. Configuration de la racine de documentation de l'hôte virtuel du serveur Web Apache

<Hôte virtuel www.joes-hardware.com>

  Nom du serveur www.joes-hardware.com

  DocumentRoot /docs/joe

Étape 4 : Cartographie et accès aux ressources

Exemple 5-3. Configuration de la racine de documentation de l'hôte virtuel du serveur Web Apache (suite)

```
  TransferLog /logs/joe.access_log
  Journal des erreurs /logs/joe.error_log
</VirtualHost>
<Hôte virtuel www.marys-antiques.com>
  Nom du serveur www.marys-antiques.com
  DocumentRoot /docs/mary
  Journal de transfert /logs/mary.access_log
  Journal des erreurs /logs/mary.error_log
</VirtualHost> ...
```

Consultez la section « Hébergement virtuel » du chapitre 18 pour plus de détails sur l'hébergement virtuel.

Répertoire personnel de l'utilisateur docroots

Une autre utilisation courante des docroots permet aux utilisateurs de créer des sites web privés sur un serveur web. Une convention courante consiste à associer les URI dont le chemin commence par une barre oblique et un tilde (/~) suivis d'un nom d'utilisateur à une racine de document privée pour cet utilisateur. Cette racine de document privée est souvent le dossier public_html dans le répertoire personnel de cet utilisateur, mais sa configuration peut être différente (Figure 5-10).

Figure 5-10. Différents docroots pour différents utilisateurs

Listes d'annuaires

Un serveur web peut recevoir des requêtes d'URL de répertoire, dont le chemin renvoie vers un répertoire, et non vers un fichier. La plupart des serveurs web peuvent être configurés pour effectuer différentes actions lorsqu'un client demande une URL de répertoire :

• Renvoyer une erreur.

• Renvoyer un « fichier d'index » spécial par défaut au lieu du répertoire.

• Analysez le répertoire et renvoyez une page HTML contenant le contenu.

La plupart des serveurs web recherchent un fichier nommé index.html ou index.htm dans un répertoire pour le représenter. Si un utilisateur demande l'URL d'un répertoire et que celui-ci contient un fichier

nommé index.html (ou index.htm), le serveur renvoie le contenu de ce fichier.

Sur le serveur web Apache, vous pouvez configurer l'ensemble des noms de fichiers qui seront interprétés comme fichiers de répertoire par défaut à l'aide de la directive de configuration DirectoryIndex. Cette directive répertorie tous les noms de fichiers servant d'index de répertoire, par ordre de préférence. La ligne de configuration suivante permet à Apache de rechercher dans un répertoire n'importe lequel des fichiers listés en réponse à une requête d'URL de répertoire :

Index du répertoire index.html index.htm home.html home.htm index.cgi

Si aucun fichier d'index par défaut n'est présent lorsqu'un utilisateur demande l'URI d'un répertoire, et si les index de répertoire ne sont pas désactivés, de nombreux serveurs web renvoient automatiquement un fichier HTML listant les fichiers de ce répertoire, leur taille et leur date de modification, ainsi que les liens URI vers chaque fichier. Cette liste de fichiers peut être pratique, mais elle permet également aux personnes curieuses de trouver sur un serveur web des fichiers qu'elles ne trouveraient pas normalement.

Vous pouvez désactiver la génération automatique des fichiers d'index de répertoire avec Apache

directif:

Options - Index

Cartographie des ressources de contenu dynamique

Les serveurs web peuvent également associer des URI à des ressources dynamiques, c'est-à-dire à des programmes générant du contenu à la demande (Figure 5-11). En fait, une classe entière de serveurs web, appelés serveurs d'applications, les connecte à des applications back-end sophistiquées. Le serveur web doit être capable de déterminer si une ressource est dynamique, où se trouve le programme générateur de contenu dynamique et comment l'exécuter. La plupart des serveurs web fournissent des mécanismes de base pour identifier et associer les ressources dynamiques.

Apache permet de mapper des composants de chemin d'accès URI vers des répertoires de programmes exécutables. Lorsqu'un serveur reçoit une requête pour un URI avec un composant de chemin d'accès exécutable, il tente d'exécuter un programme dans le répertoire serveur correspondant. Par exemple, la directive de configuration Apache suivante spécifie que tous les URI dont le chemin commence par /cgi-bin/ doivent exécuter les programmes correspondants situés dans le répertoire /usr/local/etc/httpd/cgi-programs/ :

ScriptAlias /cgi-bin/ /usr/local/etc/httpd/cgi-programs/

Apache permet également de marquer les fichiers exécutables avec une extension spécifique. Ainsi, les scripts exécutables peuvent être placés dans n'importe quel répertoire. La directive de configuration Apache suivante spécifie que toutes les ressources web se terminant par .cgi doivent être exécutées :

script cgi AddHandler .cgi

CGI est une interface ancienne, simple et populaire pour l'exécution d'applications côté serveur. Les serveurs d'applications modernes offrent une prise en charge plus puissante et plus efficace du contenu dynamique côté serveur, notamment les pages Active Server de Microsoft et les servlets Java.

Étape 4 : Cartographie et accès aux ressources

Figure 5-11. Un serveur Web peut gérer des ressources statiques et dynamiques.

Inclusions côté serveur (SSI)

De nombreux serveurs Web prennent également en charge les inclusions côté serveur. Si une ressource est signalée comme contenant des inclusions côté serveur, le serveur traite le contenu de la ressource avant de l'envoyer au client.

Le contenu est analysé à la recherche de modèles spécifiques (souvent contenus dans des commentaires HTML spécifiques), qui peuvent être des noms de variables ou des scripts intégrés. Ces modèles spécifiques sont remplacés par les valeurs de variables ou le résultat de scripts exécutables. C'est un moyen simple de créer du contenu dynamique.

Contrôles d'accès

Les serveurs web peuvent également attribuer des contrôles d'accès à des ressources spécifiques. Lorsqu'une requête arrive pour une ressource à accès contrôlé, le serveur web peut contrôler l'accès en fonction de l'adresse IP du client, ou émettre un mot de passe pour obtenir l'accès.

ressource.

Reportez-vous au chapitre 12 pour plus d’informations sur l’authentification HTTP.

Étape 5 : Élaborer des réponses

Une fois la ressource identifiée, le serveur Web exécute l'action décrite dans la méthode de requête et renvoie le message de réponse. Ce message contient un code d'état, des en-têtes et, le cas échéant, un corps de réponse. Les codes de réponse HTTP sont détaillés dans la section « Codes d'état » du chapitre 3.

Entités de réponse

Si la transaction a généré un corps de réponse, son contenu est renvoyé avec le message de réponse. S'il y a un corps, le message de réponse contient généralement :

• Un en-tête Content-Type, décrivant le type MIME du corps de la réponse

• Un en-tête Content-Length, décrivant la taille du corps de la réponse

• Le contenu réel du corps du message

Saisie MIME

Le serveur web est chargé de déterminer le type MIME du corps de la réponse. Il existe plusieurs façons de configurer les serveurs pour associer les types MIME aux ressources :

types.mime

Le serveur Web peut utiliser l'extension du nom de fichier pour indiquer le type MIME. Il analyse un fichier contenant les types MIME de chaque extension afin de calculer le type MIME de chaque ressource. Cette association de type basée sur l'extension est la plus courante ; elle est illustrée à la figure 5-12.

Figure 5-12. Un serveur Web utilise un fichier de types MIME pour définir le type de contenu sortant des ressources.

Étape 5 : Élaborer des réponses

Dactylographie magique

Le serveur web Apache peut analyser le contenu de chaque ressource et le comparer à une table de modèles connus (appelée fichier magique) afin de déterminer le type MIME de chaque fichier. Cette méthode peut être lente, mais pratique, surtout si les fichiers sont nommés sans extensions standard.

Saisie explicite

Les serveurs Web peuvent être configurés pour forcer certains fichiers ou contenus de répertoires à avoir un type MIME, quelle que soit l'extension du fichier ou le contenu.

Négociation de type

Certains serveurs web peuvent être configurés pour stocker une ressource dans plusieurs formats de document. Dans ce cas, le serveur web peut être configuré pour déterminer le format optimal à utiliser (et le type MIME associé) par un processus de négociation avec l'utilisateur. Nous aborderons ce sujet au chapitre 17.

Les serveurs Web peuvent également être configurés pour associer des fichiers particuliers à des types MIME.

Redirection

Les serveurs web renvoient parfois des réponses de redirection plutôt que des messages de réussite. Un serveur web peut rediriger le navigateur vers un autre emplacement pour exécuter la requête. Une réponse de redirection est indiquée par un code de retour 3XX. L'en-tête de réponse Location contient un URI pour le nouvel emplacement ou l'emplacement préféré du contenu. Les redirections sont utiles pour :

Ressources déplacées de façon permanente

Une ressource peut avoir été déplacée vers un nouvel emplacement ou renommée, ce qui lui a donné une nouvelle URL. Le serveur web peut informer le client que la ressource a été renommée, et le client peut mettre à jour ses signets, etc., avant de récupérer la ressource depuis son nouvel emplacement. Le code d'état 301 « Déplacé définitivement » est utilisé pour ce type de redirection.

Ressources temporairement déplacées

Si une ressource est temporairement déplacée ou renommée, le serveur peut souhaiter rediriger le client vers le nouvel emplacement. Cependant, comme le changement de nom est temporaire, le serveur souhaite que le client revienne ultérieurement avec l'ancienne URL et ne mette pas à jour ses signets. Les codes d'état 303 Voir autre et 307 Redirection temporaire sont utilisés pour ce type de redirection.

Augmentation d'URL

Les serveurs utilisent souvent des redirections pour réécrire les URL, souvent pour intégrer du contexte. À l'arrivée de la requête, le serveur génère une nouvelle URL contenant des informations d'état intégrées et redirige l'utilisateur vers cette nouvelle URL.* Le client suit la redirection et réémet la requête, mais en incluant désormais l'URL complète avec l'état augmenté. Il s'agit d'un

* Ces URL étendues et augmentées par l’état sont parfois appelées « URL lourdes ».

Moyen utile de maintenir l'état des transactions. Les codes d'état 303 « Voir autre » et 307 « Redirection temporaire » sont utilisés pour ce type de redirection.

Équilibrage de charge

Si un serveur surchargé reçoit une requête, il peut rediriger le client vers un serveur moins chargé. Les codes d'état 303 Voir autre et 307 Redirection temporaire sont utilisés pour ce type de redirection.

Affinité du serveur

Les serveurs Web peuvent disposer d'informations locales sur certains utilisateurs ; un serveur peut rediriger le client vers un serveur contenant des informations le concernant. Les codes d'état 303 Voir autre et 307 Redirection temporaire sont utilisés pour ce type de redirection.

Canonisation des noms de répertoires

Lorsqu'un client demande un URI pour un nom de répertoire sans barre oblique finale, la plupart des serveurs Web redirigent le client vers un URI avec la barre oblique ajoutée, afin que les liens relatifs fonctionnent correctement.

Étape 6 : Envoi des réponses

Les serveurs Web rencontrent des problèmes similaires pour l'envoi et la réception de données entre les connexions. Le serveur peut avoir de nombreuses connexions vers de nombreux clients : certaines sont inactives, d'autres envoient des données au serveur et d'autres encore renvoient les données de réponse aux clients.

Le serveur doit surveiller l'état de la connexion et gérer les connexions persistantes avec une attention particulière. Pour les connexions non persistantes, le serveur doit fermer sa partie de la connexion une fois le message envoyé.

Pour les connexions persistantes, la connexion peut rester ouverte, auquel cas le serveur doit être très prudent pour calculer correctement l'en-tête Content-Length, sinon le client n'aura aucun moyen de savoir quand une réponse se termine (voir chapitre 4).

Étape 7 : Journalisation

Enfin, une fois la transaction terminée, le serveur web enregistre une entrée dans un fichier journal décrivant la transaction effectuée. La plupart des serveurs web proposent plusieurs formes de journalisation configurables. Consultez le chapitre 21 pour plus de détails.

Pour plus d'informations

Pour plus d'informations sur Apache, Jigsaw et ident, consultez :

Apache : le guide définitif

Ben Laurie et Peter Laurie, O'Reilly & Associates, Inc.

Apache professionnel

Pierre Wainwright, Wrox Press.

Pour plus d'informations

http://www.w3c.org/Jigsaw/

Jigsaw — Serveur du W3C Site Web du Consortium du W3C.
http://www.ietf.org/rfc/rfc1413.txt

RFC 1413, « Protocole d'identification », par M. St. Johns.

Chapitre 6Ceci est le titre du livre CHAPITRE 6

Procurations

Les serveurs proxy web sont des intermédiaires. Ils se situent entre les clients et les serveurs et agissent comme des intermédiaires, acheminant les messages HTTP entre les parties. Ce chapitre aborde les serveurs proxy HTTP, la prise en charge spécifique des fonctionnalités proxy et certains comportements complexes rencontrés lors de leur utilisation.

Dans ce chapitre, nous :

• Expliquez les proxys HTTP, en les comparant aux passerelles Web et en illustrant comment les proxys sont déployés.

• Montrez quelques-unes des façons dont les proxys sont utiles.

• Décrivez comment les proxys sont déployés dans les réseaux réels et comment le trafic est dirigé vers les serveurs proxy.

• Montrez comment configurer votre navigateur pour utiliser un proxy.

• Démontrer les requêtes proxy HTTP, en quoi elles diffèrent des requêtes serveur et comment les proxys peuvent subtilement modifier le comportement des navigateurs.

• Expliquez comment vous pouvez enregistrer le chemin de vos messages à travers des chaînes de serveurs proxy, en utilisant les en-têtes Via et la méthode TRACE.

• Décrire le contrôle d'accès HTTP basé sur un proxy.

• Expliquez comment les proxys peuvent interagir entre les clients et les serveurs, chacun pouvant prendre en charge différentes fonctionnalités et versions.

Intermédiaires Web

Les serveurs proxy web sont des intermédiaires qui effectuent les transactions pour le compte du client. Sans proxy web, les clients HTTP communiquent directement avec les serveurs HTTP. Avec un proxy web, le client communique avec le proxy, qui communique lui-même avec le serveur pour le compte du client. Le client finalise la transaction, mais grâce aux services du serveur proxy.

Les serveurs proxy HTTP sont à la fois des serveurs et des clients web. Comme les clients HTTP envoient des requêtes aux proxys, le serveur proxy doit gérer correctement les requêtes et les connexions, et renvoyer des réponses, comme un serveur web. Parallèlement, le proxy

envoie des requêtes aux serveurs ; il doit donc se comporter comme un client HTTP, envoyant des requêtes et recevant des réponses (voir Figure 6-1). Si vous créez votre propre proxy HTTP, vous devrez respecter scrupuleusement les règles applicables aux clients et aux serveurs HTTP.

Figure 6-1. Un proxy doit être à la fois serveur et client.

Proxys privés et partagés

Un serveur proxy peut être dédié à un seul client ou partagé entre plusieurs clients. Les proxys dédiés à un seul client sont appelés proxys privés. Les proxys partagés entre plusieurs clients sont appelés proxys publics.

Procurations publiques

La plupart des proxys sont publics et partagés. Un proxy centralisé est plus économique et plus simple à administrer. Certaines applications proxy, comme les serveurs proxy de mise en cache, gagnent en utilité à mesure que davantage d'utilisateurs sont redirigés vers le même serveur proxy, car elles peuvent exploiter les requêtes communes entre utilisateurs.

Procurations privées

Les proxys privés dédiés sont moins courants, mais ils ont leur place, notamment lorsqu'ils sont exécutés directement sur l'ordinateur client. Certains assistants de navigation, ainsi que certains services FAI, exécutent de petits proxys directement sur le PC de l'utilisateur afin d'étendre les fonctionnalités du navigateur, d'améliorer les performances ou d'héberger des publicités pour des services FAI gratuits.

Proxies contre passerelles

À proprement parler, les proxys connectent deux applications ou plus utilisant le même protocole, tandis que les passerelles connectent deux parties ou plus utilisant des protocoles différents. Une passerelle agit comme un « convertisseur de protocole », permettant à un client d'effectuer une transaction avec un serveur, même lorsque le client et le serveur utilisent des protocoles différents.

La figure 6-2 illustre la différence entre les proxys et les passerelles :

• Le périphérique intermédiaire de la figure 6-2a est un proxy HTTP, car le proxy parle HTTP à la fois au client et au serveur.

• Le périphérique intermédiaire de la figure 6-2b est une passerelle HTTP/POP, car il relie une interface HTTP à un serveur de messagerie POP. La passerelle convertit les transactions Web en transactions POP appropriées, permettant ainsi à l'utilisateur de lire ses e-mails via HTTP.

Les logiciels de messagerie Web tels que Yahoo! Mail et MSN Hotmail sont des passerelles HTTP.

Figure 6-2. Les proxys utilisent le même protocole ; les passerelles relient différents protocoles.

En pratique, la différence entre proxys et passerelles est floue. Comme les navigateurs et les serveurs implémentent différentes versions de HTTP, les proxys effectuent souvent une certaine conversion de protocole. Les serveurs proxy commerciaux implémentent des fonctionnalités de passerelle pour prendre en charge les protocoles de sécurité SSL, les pare-feu SOCKS, l'accès FTP et les applications web. Nous reviendrons sur les passerelles au chapitre 8.

Pourquoi utiliser des proxys ?

Les serveurs proxy offrent de nombreuses possibilités pratiques et astucieuses. Ils peuvent améliorer la sécurité, optimiser les performances et réaliser des économies. De plus, comme ils peuvent voir et analyser l'intégralité du trafic HTTP, les proxys peuvent surveiller et modifier ce trafic afin de mettre en œuvre de nombreux services web à valeur ajoutée. Voici quelques exemples d'utilisation des proxys :

Filtre enfant (Figure 6-3)

Les écoles primaires utilisent des proxys de filtrage pour bloquer l'accès aux contenus réservés aux adultes, tout en offrant un accès libre aux sites éducatifs. Comme le montre la figure 6-3, le proxy peut autoriser un accès illimité aux contenus éducatifs, mais interdire l'accès aux sites inappropriés pour les enfants.*

Figure 6-3. Exemple d'application proxy : filtre Internet sécurisé pour les enfants

Contrôleur d'accès aux documents (Figure 6-4)

Les serveurs proxy permettent de mettre en œuvre une stratégie de contrôle d'accès uniforme sur un vaste ensemble de serveurs et de ressources web, et de créer une piste d'audit. Ceci est utile dans les grandes entreprises ou autres bureaucraties décentralisées.

Tous les contrôles d'accès peuvent être configurés sur le serveur proxy centralisé, sans nécessiter de mise à jour fréquente des contrôles d'accès sur de nombreux sites Web.

serveurs, de différentes marques et modèles, administrés par différentes organisations.† Dans la figure 6-4, le proxy de contrôle d'accès centralisé :

• Permet au client 1 d'accéder aux pages d'actualités du serveur A sans restriction

• Donne au client 2 un accès illimité au contenu Internet

• Nécessite un mot de passe du client 3 avant d'autoriser l'accès au serveur B

Pare-feu de sécurité (Figure 6-5)

Les ingénieurs en sécurité réseau utilisent souvent des serveurs proxy pour renforcer la sécurité. Ces serveurs limitent les flux de protocoles applicatifs entrants et sortants d'une organisation, en un seul point sécurisé du réseau. Ils peuvent également fournir des points d'ancrage pour analyser ce trafic (Figure 6-5), comme c'est le cas pour les sites web et les e-mails antivirus.

procurations.

* Plusieurs entreprises et organisations à but non lucratif fournissent des logiciels de filtrage et maintiennent des « listes noires » afin d'identifier et de restreindre l'accès aux contenus répréhensibles.

† Pour empêcher les utilisateurs avertis de contourner volontairement le proxy de contrôle, les serveurs Web peuvent être configurés de manière statique pour accepter uniquement les demandes provenant des serveurs proxy.

Figure 6-5. Exemple d'application proxy : pare-feu de sécurité

Cache Web (Figure 6-6)

Les caches proxy conservent des copies locales des documents populaires et les diffusent à la demande, réduisant ainsi la lenteur et le coût des communications Internet.

Dans la figure 6-6, les clients 1 et 2 accèdent à l'objet A à partir d'un cache Web à proximité, tandis que les clients 3 et 4 accèdent au document à partir du serveur d'origine.

Figure 6-6. Exemple d'application proxy : cache Web

Mère porteuse (Figure 6-7)

Les proxys peuvent se faire passer pour des serveurs web. Ces proxys, appelés « substituts » ou « proxies inverses », reçoivent de véritables requêtes de serveurs web, mais, contrairement aux serveurs web, ils peuvent initier des communications avec d'autres serveurs pour localiser le contenu demandé à la demande.

Les substituts peuvent être utilisés pour améliorer les performances des serveurs web lents pour le contenu courant. Dans cette configuration, les substituts sont souvent appelés accélérateurs de serveur (Figure 6-7). Ils peuvent également être utilisés en conjonction avec la fonctionnalité de routage de contenu pour créer des réseaux distribués de contenu répliqué à la demande.

Figure 6-7. Exemple d'application proxy : surrogate (dans un déploiement d'accélérateur de serveur)

Routeur de contenu (Figure 6-8)

Les serveurs proxy peuvent agir comme des « routeurs de contenu », en acheminant les requêtes vers des serveurs Web particuliers en fonction des conditions de trafic Internet et du type de contenu.

Les routeurs de contenu peuvent également être utilisés pour implémenter diverses offres de niveau de service. Par exemple, ils peuvent transférer les requêtes vers des caches de réplication proches si l'utilisateur ou le fournisseur de contenu a souscrit à un service de performances supérieures (Figure 6-8), ou acheminer les requêtes HTTP via des proxys de filtrage si l'utilisateur a souscrit à un service de filtrage. De nombreux services intéressants peuvent être créés grâce à des proxys de routage de contenu adaptatifs.

Figure 6-8. Exemple d'application proxy : routage de contenu

Transcodeur (Figure 6-9)

Les serveurs proxy peuvent modifier le format du contenu avant de le transmettre aux clients. Cette traduction transparente entre les représentations de données est appelée transcodage.*

Les proxys de transcodage peuvent convertir les images GIF en JPEG au fur et à mesure de leur défilement, afin d'en réduire la taille. Les images peuvent également être réduites et leur intensité de couleur atténuée pour être lisibles sur un téléviseur. De même, les fichiers texte peuvent être compressés et de courts résumés de pages web peuvent être générés pour les pagers connectés et les smartphones. Les proxys peuvent même convertir des documents en langues étrangères à la volée !

La figure 6-9 montre un proxy de transcodage qui convertit le texte anglais en texte espagnol et reformate également les pages HTML en texte plus simple qui peut être affiché sur le petit écran d'un téléphone mobile.

* Certains distinguent « transcodage » et « traduction », définissant le transcodage comme une conversion relativement simple de l'encodage

des données (par exemple, une compression sans perte) et la traduction comme un reformatage ou des modifications sémantiques plus importantes des données. Nous utilisons le terme « transcodage » pour désigner toute modification intermédiaire du contenu.

Figure 6-9. Exemple d'application proxy : transcodeur de contenu

Anonymiseur (Figure 6-10)

Les proxys anonymiseurs offrent une confidentialité et un anonymat accrus, en supprimant activement les caractéristiques d'identification des messages HTTP (par exemple, l'adresse IP du client, l'en-tête De, l'en-tête Referer, les cookies, les identifiants de session URI).*

Dans la figure 6-10, le proxy anonymisant apporte les modifications suivantes aux messages de l'utilisateur pour augmenter la confidentialité :

• L'ordinateur et le type de système d'exploitation de l'utilisateur sont supprimés de l'en-tête User-Agent.

• L'en-tête De est supprimé pour protéger l'adresse e-mail de l'utilisateur.

• L'en-tête Referer est supprimé pour masquer les autres sites visités par l'utilisateur.

• Les en-têtes de cookies sont supprimés pour éliminer les données de profilage et d’identité.

Figure 6-10. Exemple d'application proxy : anonymiseur

* Cependant, étant donné que les informations d'identification sont supprimées, la qualité de l'expérience de navigation de l'utilisateur peut être diminuée et certains sites Web peuvent ne pas fonctionner correctement.

Où vont les proxys ?

La section précédente a expliqué le rôle des proxys. Voyons maintenant où ils se situent lorsqu'ils sont déployés dans une architecture réseau. Nous aborderons :

• Comment les proxys peuvent être déployés dans les réseaux

• Comment les proxys peuvent s'enchaîner dans des hiérarchies

• Comment le trafic est dirigé vers un serveur proxy en premier lieu

Déploiement du serveur proxy

Vous pouvez placer des proxys dans toutes sortes d’endroits, en fonction de leurs utilisations prévues.

La figure 6-11 illustre quelques façons dont les serveurs proxy peuvent être déployés.

Proxy de sortie (Figure 6-11a)

Vous pouvez installer des proxys aux points de sortie des réseaux locaux pour contrôler le flux de trafic entre le réseau local et l'Internet. Dans une entreprise, vous pouvez utiliser des proxys de sortie pour offrir une protection pare-feu contre les pirates informatiques externes ou pour réduire les coûts de bande passante et améliorer les performances du trafic Internet. Une école primaire pourrait utiliser un proxy de sortie filtrant pour empêcher les élèves précoces de consulter des contenus inappropriés.

Proxy d'accès (entrée) (Figure 6-11b)

Les proxys sont souvent placés aux points d'accès des FAI et traitent l'ensemble des requêtes des clients. Les FAI utilisent des proxys de mise en cache pour stocker des copies de documents populaires, améliorer la vitesse de téléchargement pour leurs utilisateurs (notamment ceux disposant de connexions haut débit) et réduire les coûts de bande passante Internet.

Mères porteuses (Figure 6-11c)

Les proxys sont souvent déployés comme des substituts (aussi appelés proxys inverses) à la périphérie du réseau, devant les serveurs web. Ils peuvent ainsi traiter toutes les requêtes adressées au serveur web et lui demander des ressources uniquement lorsque cela est nécessaire. Les substituts peuvent renforcer la sécurité des serveurs web ou améliorer les performances en plaçant des caches rapides devant les serveurs plus lents. Ils utilisent généralement le nom et l'adresse IP du serveur web, de sorte que toutes les requêtes sont transmises au proxy plutôt qu'au serveur.

Proxy d'échange réseau (Figure 6-11d)

Avec une puissance suffisante, les proxys peuvent être placés dans les points d'échange de peering Internet entre les réseaux, pour réduire la congestion aux jonctions Internet grâce à la mise en cache et pour surveiller les flux de trafic.*

* Les proxys centraux sont souvent déployés là où la bande passante Internet est très coûteuse (notamment en Europe). Certains pays (comme le Royaume-Uni) envisagent également des déploiements de proxy controversés pour surveiller le trafic Internet, pour des raisons de sécurité nationale.

Où vont les proxys ?

Figure 6-11. Les proxys peuvent être déployés de différentes manières, selon leur utilisation prévue.

Hiérarchies de proxy

Les proxys peuvent être organisés en cascade dans des chaînes appelées hiérarchies de proxys. Dans une hiérarchie de proxys, les messages sont transmis de proxy en proxy jusqu'à atteindre le serveur d'origine (puis sont retransmis au client via les proxys), comme illustré à la figure 6-12.

Dans une hiérarchie de proxys, les serveurs proxy se voient attribuer des relations parent-enfant. Le proxy entrant suivant (le plus proche du serveur) est appelé parent, et le proxy sortant suivant (le plus proche du client) est appelé enfant. Dans la figure 6-12, le proxy 1 est l'enfant.

Figure 6-12. Hiérarchie proxy à trois niveaux

proxy du proxy 2. De même, le proxy 2 est le proxy enfant du proxy 3, et le proxy 3 est le proxy parent du proxy 2.

Routage du contenu de la hiérarchie proxy

La hiérarchie des proxys illustrée à la figure 6-12 est statique : le proxy 1 transmet toujours les messages au proxy 2, et ce dernier les transmet toujours au proxy 3. Cependant, les hiérarchies ne sont pas nécessairement statiques. Un serveur proxy peut transmettre des messages à un ensemble varié et évolutif de serveurs proxy et de serveurs d'origine, en fonction de nombreux facteurs.

Par exemple, dans la figure 6-13, le proxy d'accès achemine vers des proxys parents ou des serveurs d'origine dans différentes circonstances :

• Si l'objet demandé appartient à un serveur Web qui a payé pour la distribution de contenu, le proxy pourrait acheminer la demande vers un serveur de cache à proximité qui renverrait l'objet mis en cache ou le récupérerait s'il n'était pas disponible.

• Si la demande concernait un type particulier d'image, le proxy d'accès pourrait acheminer la demande vers un proxy de compression dédié qui récupérerait l'image puis la compresserait, afin qu'elle soit téléchargée plus rapidement via un modem lent vers le client.

Figure 6-13. Les hiérarchies de proxy peuvent être dynamiques et changer à chaque requête.

Où vont les proxys ?

Voici quelques autres exemples de sélection dynamique des parents :

Équilibrage de charge

Un proxy enfant peut choisir un proxy parent en fonction du niveau actuel de charge de travail des parents, afin de répartir la charge.

Routage de proximité géographique

Un proxy enfant peut sélectionner un parent responsable de la région géographique du serveur d'origine.

Protocole/type de routage

Un proxy enfant peut router les requêtes vers différents serveurs parents et d'origine en fonction de l'URI. Certains types d'URI peuvent entraîner le transport des requêtes via des serveurs proxy spécifiques, pour un traitement de protocole spécifique.

Routage par abonnement

Si les éditeurs ont payé un supplément pour un service hautes performances, leurs URI peuvent être acheminés vers des caches volumineux ou des moteurs de compression pour améliorer les performances.

La logique de routage parental dynamique est implémentée différemment dans différents produits, notamment les fichiers de configuration, les langages de script et les plug-ins exécutables dynamiques.

Comment les proxys génèrent du trafic

Comme les clients communiquent généralement directement avec les serveurs web, il est important d'expliquer comment le trafic HTTP parvient à un proxy. Il existe quatre manières courantes de faire parvenir le trafic client à un proxy :

Modifier le client

De nombreux clients web, notamment Netscape et les navigateurs Microsoft, prennent en charge la configuration manuelle et automatique du proxy. Si un client est configuré pour utiliser un serveur proxy, il envoie intentionnellement ses requêtes HTTP directement au proxy, plutôt qu'au serveur d'origine (figure 6-14a).

Modifier le réseau

Il existe plusieurs techniques par lesquelles l'infrastructure réseau intercepte et dirige le trafic web vers un proxy, à l'insu du client et sans sa participation. Cette interception repose généralement sur des dispositifs de commutation et de routage qui surveillent le trafic HTTP,

l'interceptent et le redirigent vers un proxy, à l'insu du client (Figure 6-14b). On parle alors de proxy d'interception*.

Modifier l'espace de noms DNS

Les substituts, qui sont des serveurs proxy placés devant les serveurs Web, prennent directement le nom et l'adresse IP du serveur Web, de sorte que toutes les requêtes leur sont adressées.

* Les proxys d'interception sont communément appelés « proxies transparents », car vous vous y connectez sans vous rendre compte de leur présence. Le terme « transparence » étant déjà utilisé dans les spécifications HTTP pour désigner les fonctions qui ne modifient pas le comportement sémantique, la communauté des standards suggère d'utiliser le terme « interception » pour la capture du trafic. Nous adoptons cette nomenclature ici.

du serveur (Figure 6-14c). Cela peut être réalisé en modifiant manuellement les tables de nommage DNS ou en utilisant des serveurs DNS dynamiques spécifiques qui calculent le proxy ou le serveur approprié à utiliser à la demande. Dans certaines installations, l'adresse IP et le nom du serveur réel sont modifiés, et le serveur de substitution reçoit l'ancienne adresse et le nouveau nom.

Modifier le serveur Web

Certains serveurs Web peuvent également être configurés pour rediriger les demandes des clients vers un proxy en renvoyant une commande de redirection HTTP (code de réponse 305) au client.

Après avoir reçu la redirection, le client effectue une transaction avec le proxy (Figure 6-14d).

La section suivante explique comment configurer les clients pour envoyer du trafic vers des proxys. Le chapitre 20 explique comment configurer le réseau, le DNS et les serveurs pour rediriger le trafic vers des serveurs proxy.

Figure 6-14. Il existe de nombreuses techniques pour diriger les requêtes web vers des proxys.

Paramètres du proxy client

Tous les navigateurs web modernes permettent de configurer l'utilisation de proxys. De nombreux navigateurs proposent d'ailleurs plusieurs méthodes de configuration, notamment :

Configuration manuelle

Vous définissez explicitement un proxy à utiliser.

Préconfiguration du navigateur

Le fournisseur ou le distributeur du navigateur préconfigure manuellement le paramètre proxy du navigateur (ou de tout autre client Web) avant de le livrer aux clients.

Paramètres du proxy client

Configuration automatique du proxy (PAC)

Vous fournissez un URI à un fichier de configuration automatique de proxy JavaScript (PAC) ; le client récupère le fichier JavaScript et l'exécute pour décider s'il doit utiliser un proxy et, si oui, quel serveur proxy utiliser.

Découverte de proxy WPAD

Certains navigateurs prennent en charge le protocole Web Proxy Autodiscovery Protocol (WPAD), qui détecte automatiquement un « serveur de configuration » à partir duquel le navigateur peut télécharger un fichier de configuration automatique.*

Configuration du proxy client : manuelle

De nombreux clients web permettent de configurer manuellement les proxys. Netscape Navigator et Microsoft Internet Explorer offrent tous deux une prise en charge pratique de la configuration des proxys.

Dans Netscape Navigator 6, vous spécifiez les proxys via la sélection de menu Édition → Préférences → Avancé → Proxies, puis en sélectionnant le bouton radio « Configuration manuelle du proxy ».

Dans Microsoft Internet Explorer 5, vous pouvez spécifier manuellement les proxys à partir du menu Outils → Options Internet, en sélectionnant une connexion, en appuyant sur « Paramètres », en cochant la case « Utiliser un serveur proxy » et en cliquant sur « Avancé ».

D'autres navigateurs proposent différentes méthodes pour modifier manuellement la configuration, mais le principe reste le même : spécifier l'hôte et le port du proxy. Plusieurs FAI fournissent à leurs clients des navigateurs préconfigurés, ou des systèmes d'exploitation personnalisés, qui redirigent le trafic web vers des serveurs proxy.

Configuration du proxy client : fichiers PAC

La configuration manuelle du proxy est simple, mais peu flexible. Vous ne pouvez spécifier qu'un seul serveur proxy pour l'ensemble du contenu, et le basculement n'est pas pris en charge. La configuration manuelle du proxy entraîne également des problèmes administratifs pour les grandes organisations. Avec un grand nombre de navigateurs configurés, il est difficile, voire impossible, de reconfigurer chaque navigateur si des modifications sont nécessaires.

Les fichiers de configuration automatique de proxy (PAC) constituent une solution plus dynamique pour la configuration de proxy, car il s'agit

de petits programmes JavaScript qui calculent les paramètres de proxy à la volée. À chaque accès à un document, une fonction JavaScript sélectionne le serveur proxy approprié.

Pour utiliser les fichiers PAC, configurez votre navigateur avec l'URI du fichier PAC JavaScript (la configuration est similaire à la configuration manuelle, mais vous devez fournir l'URI dans une case « Configuration automatique »). Le navigateur récupérera le fichier PAC depuis cet URI et l'utilisera.

* Actuellement pris en charge uniquement par Internet Explorer.

La logique JavaScript permet de calculer le serveur proxy approprié pour chaque accès. Les fichiers PAC ont généralement un suffixe .pac et le type MIME « application/x-ns-proxy-autoconfig ».

Chaque fichier PAC doit définir une fonction appelée FindProxyForURL(url,host) qui calcule le serveur proxy approprié à utiliser pour accéder à l'URI. La valeur de retour de la fonction peut être l'une des valeurs du tableau 6-1.

Tableau 6-1. Valeurs de retour du script de configuration automatique du proxy

Valeur de retour de FindProxyForURL Description

Les connexions DIRECTES doivent être effectuées directement, sans aucun proxy.

PROXY host:port Le proxy spécifié doit être utilisé.

Hôte SOCKS : port Le serveur SOCKS spécifié doit être utilisé.

Le fichier PAC de l'exemple 6-1 requiert un proxy pour les transactions HTTP, un autre proxy pour les transactions FTP et des connexions directes pour tous les autres types de transactions.

Exemple 6-1. Exemple de fichier de configuration automatique de proxy

```
fonction FindProxyForURL(url, hôte) { si (url.substring(0,5) == "http:") {
      retourner "PROXY http-proxy.mydomain.com:8080";
   } else if (url.substring(0,4) =="ftp:") { return "PROXY ftp-
proxy.mydomain.com:8080";
   } autre {
      retourner "DIRECT";
   }
}
```

Pour plus de détails sur les fichiers PAC, reportez-vous au chapitre 20.

Configuration du proxy client : WPAD

Un autre mécanisme de configuration du navigateur est le protocole WPAD (Web Proxy Autodiscovery Protocol). WPAD est un algorithme qui utilise une stratégie progressive de mécanismes de découverte pour trouver automatiquement le fichier PAC approprié au navigateur. Un client implémentant le protocole WPAD :

• Utilisez WPAD pour trouver l'URI PAC.

• Récupérez le fichier PAC en fonction de l'URI.

• Exécutez le fichier PAC pour déterminer le serveur proxy.

• Utiliser le serveur proxy pour les requêtes.

WPAD utilise une série de techniques de découverte de ressources pour déterminer le fichier PAC approprié. Plusieurs techniques de découverte sont utilisées, car toutes les organisations ne peuvent pas les utiliser toutes. WPAD teste chaque technique, une par une, jusqu'à ce qu'elle réussisse.

Paramètres du proxy client

La spécification WPAD actuelle définit les techniques suivantes, dans l'ordre :

• Protocole de découverte dynamique d'hôte (DHCP)

• Protocole de localisation de service (SLP)

• Noms d'hôtes DNS bien connus

• Enregistrements DNS SRV

• URI du service DNS dans les enregistrements TXT

Pour plus d'informations, consultez le chapitre 20.

Les points délicats des requêtes proxy

Cette section explique certains des aspects délicats et souvent mal compris des requêtes de serveur proxy, notamment :

• En quoi les URI dans les requêtes proxy diffèrent des requêtes serveur

• Comment l'interception et les proxys inverses peuvent masquer les informations sur l'hôte du serveur

• Les règles de modification des URI

• Comment les proxys impactent les fonctionnalités intelligentes de saisie semi-automatique des URI ou d'extension du nom d'hôte d'un navigateur

Les URI proxy diffèrent des URI serveur

Les messages du serveur Web et du proxy Web ont la même syntaxe, à une exception près : l'URI d'une requête HTTP diffère lorsqu'un client envoie la requête à un serveur plutôt qu'à un proxy.

Lorsqu'un client envoie une requête à un serveur Web, la ligne de requête ne contient qu'un URI partiel (sans schéma, hôte ou port), comme illustré dans l'exemple suivant :

OBTENIR /index.html HTTP/1.0

Agent utilisateur : SuperBrowserv1.3

Cependant, lorsqu'un client envoie une requête à un proxy, la ligne de requête contient l'URI complète. Par exemple :

OBTENIR http://www.marys-antiques.com/index.html HTTP/1.0

Agent utilisateur : SuperBrowser v1.3

Pourquoi deux formats de requête différents, un pour les proxys et un pour les serveurs ? Dans la conception HTTP d'origine, les clients communiquaient directement avec un seul serveur. L'hébergement virtuel n'existait pas et aucune disposition n'était prévue pour les proxys. Comme un seul serveur connaît son propre nom d'hôte et son propre port, pour éviter l'envoi d'informations redondantes, les clients n'envoyaient que l'URI partielle, sans le schéma, l'hôte (et le port).

Avec l'apparition des proxys, les URI partiels sont devenus problématiques. Les proxys devaient connaître le nom du serveur de destination afin de pouvoir établir leurs propres connexions.

Le serveur. Les passerelles proxy avaient besoin du schéma d'URI pour se connecter aux ressources FTP et autres schémas. HTTP/1.0 a résolu le problème en exigeant l'URI complet pour les requêtes proxy, mais en conservant des URI partiels pour les requêtes serveur (le nombre de serveurs déjà déployés était trop important pour tous les modifier afin de prendre en charge les URI complets).*

Nous devons donc envoyer des URI partiels aux serveurs et des URI complets aux proxys. Si les paramètres du proxy client sont explicitement configurés, le client sait quel type de requête émettre :

• Lorsque le client n’est pas configuré pour utiliser un proxy, il envoie l’URI partielle (Figure 6-15a).

• Lorsque le client est configuré pour utiliser un proxy, il envoie l'URI complet (Figure 6-15b).

Figure 6-15. Les proxys intercepteurs recevront les requêtes du serveur.

* HTTP/1.1 exige désormais que les serveurs gèrent les URI complets pour les requêtes proxy et serveur, mais dans la pratique, de nombreux serveurs déployés n'acceptent encore que des URI partiels.

Le même problème avec l'hébergement virtuel

Le problème de « schéma/hôte/port manquant » du proxy est le même que celui rencontré par les serveurs web hébergés virtuellement. Ces serveurs partagent le même serveur web physique avec plusieurs sites web. Lorsqu'une requête arrive pour l'URI partielle /index.html, le serveur web hébergé virtuellement doit connaître le nom d'hôte du site web visé (voir « Doccroots hébergés virtuellement » au chapitre 5 et « Hébergement virtuel » au chapitre 18 pour plus d'informations).

Bien que les problèmes soient similaires, ils ont été résolus de différentes manières :

• Les proxys explicites résolvent le problème en exigeant un URI complet dans le message de demande.

• Les serveurs Web hébergés virtuellement nécessitent un en-tête d'hôte pour transporter les informations sur l'hôte et le port.

Les proxys interceptés obtiennent des URI partiels

Tant que les clients implémentent correctement HTTP, ils enverront les URI complètes dans leurs requêtes aux proxys explicitement configurés. Cela résout une partie du problème, mais il y a un hic : un client ne sait pas toujours qu'il communique avec un proxy, car certains proxys peuvent lui être invisibles. Même si le client n'est pas configuré pour utiliser un proxy, son trafic peut transiter par un proxy de substitution ou d'interception. Dans les deux cas, le client pensera communiquer avec un serveur web et n'enverra pas l'URI complète :

• Un substitut, comme décrit précédemment, est un serveur proxy qui remplace le serveur d'origine, généralement en s'appropriant son nom d'hôte ou son adresse IP. Il reçoit la requête du serveur web et peut transmettre des réponses en cache ou des requêtes proxy au serveur réel. Un client ne peut pas distinguer un substitut d'un serveur web ; il envoie donc des URI partiels (Figure 6-15c).

• Un proxy d'interception est un serveur proxy du flux réseau qui détourne le trafic du client vers le serveur et fournit une réponse en cache ou la transmet par proxy. Comme le proxy d'interception détourne le trafic client vers serveur, il reçoit des URI partiels envoyés aux serveurs web (Figure 6-15d).*

Les proxys peuvent gérer à la fois les requêtes proxy et serveur

En raison des différentes manières de rediriger le trafic vers les serveurs proxy, les serveurs proxy génériques doivent prendre en charge les URI complets et partiels dans les messages de requête. Le proxy doit utiliser l'URI complet s'il s'agit d'une requête proxy explicite,

ou l'URI partiel et l'en-tête d'hôte virtuel s'il s'agit d'une requête de serveur web.

* Les proxys d'interception peuvent également intercepter le trafic client-proxy dans certaines circonstances. Dans ce cas, le proxy intercepteur peut obtenir des URI complets et devoir les gérer. Cela est rare, car les proxys explicites communiquent généralement sur un port différent de celui utilisé par HTTP (généralement 8080 au lieu de 80), et les proxys d'interception n'interceptent généralement que le port 80.

Les règles d'utilisation des URI complets et partiels sont les suivantes :

• Si un URI complet est fourni, le proxy doit l'utiliser.

• Si un URI partiel est fourni et qu'un en-tête Host est présent, l'en-tête Host doit être utilisé pour déterminer le nom du serveur d'origine et le numéro de port.

• Si un URI partiel est fourni et qu'il n'y a pas d'en-tête Host, le serveur d'origine doit être déterminé d'une autre manière :

— Si le proxy est un substitut, remplaçant un serveur d'origine, le proxy peut être configuré avec l'adresse et le numéro de port du serveur réel.

— Si le trafic a été intercepté et que l'intercepteur rend l'adresse IP et le port d'origine disponibles, le proxy peut utiliser l'adresse IP et le numéro de port de la technologie d'interception (voir Chapitre 20).

— Si tout le reste échoue, le proxy n'a pas suffisamment d'informations pour déterminer le serveur d'origine et doit renvoyer un message d'erreur (suggérant souvent à l'utilisateur de passer à un navigateur moderne qui prend en charge les en-têtes d'hôte).*

Modification de l'URI en vol

Les serveurs proxy doivent être très prudents lorsqu'ils modifient l'URI de la requête lors de la transmission des messages. De légères modifications de l'URI, même si elles semblent anodines, peuvent créer des problèmes d'interopérabilité avec les serveurs en aval.

En particulier, certains proxys sont connus pour « canoniser » les URI dans un format standard avant de les transmettre au saut suivant. Des transformations apparemment anodines, comme le remplacement des ports HTTP par défaut par un « :80 » explicite, ou la correction des URI en remplaçant les caractères réservés illégaux par leurs substitutions correctement échappées, peuvent entraîner des problèmes d'interopérabilité.

En général, les serveurs proxy doivent s'efforcer d'être aussi tolérants que possible. Ils ne doivent pas se comporter comme des « policiers du protocole » cherchant à faire respecter strictement le protocole, car cela pourrait entraîner une perturbation importante des services auparavant fonctionnels.

En particulier, les spécifications HTTP interdisent aux proxys d'interception généraux de réécrire les chemins absolus des URI lors de leur transmission. La seule exception est qu'ils peuvent remplacer un chemin vide par « / ».

Extension automatique du client URI et résolution du nom d'hôte

Les navigateurs résolvent les URI de requête différemment, selon la présence ou non d'un proxy. Sans proxy, le navigateur utilise l'URI que vous saisissez et tente de trouver l'adresse IP correspondante. Si le nom d'hôte est trouvé, le navigateur essaie les adresses IP correspondantes jusqu'à ce qu'une connexion soit établie.

* Cela ne doit pas être fait à la légère. Les utilisateurs recevront des pages d'erreur cryptiques qu'ils n'ont jamais vues auparavant.

Mais si l'hôte n'est pas trouvé, de nombreux navigateurs tentent de fournir une « extension » automatique des noms d'hôtes, au cas où vous auriez saisi une abréviation « abrégée » de l'hôte (reportez-vous à « URL Expandomatic » au chapitre 2) :*

• De nombreux navigateurs tentent d'ajouter un préfixe « www. » et un suffixe « .com », au cas où vous auriez simplement saisi la partie centrale d'un nom de site Web courant (par exemple, pour permettre aux utilisateurs de saisir « Yahoo » au lieu de « www.yahoo.com »).

• Certains navigateurs transmettent même votre URI non résoluble à un site tiers, qui tente de corriger les fautes d'orthographe et de suggérer les URI que vous auriez pu vouloir.

• De plus, la configuration DNS de la plupart des systèmes vous permet de saisir uniquement le préfixe du nom d'hôte, et le DNS recherche automatiquement le domaine. Si vous utilisez le domaine « oreilly.com » et saisissez le nom d'hôte « host7 », le DNS essaie automatiquement de trouver « host7.oreilly.com ». Ce n'est pas un nom d'hôte complet et valide.

Résolution d'URI sans proxy

La figure 6-16 montre un exemple d'extension automatique du nom d'hôte d'un navigateur sans proxy. Aux étapes 2a à 3c, le navigateur recherche différentes variantes du nom d'hôte jusqu'à ce qu'un nom d'hôte valide soit trouvé.

Figure 6-16. Le navigateur développe automatiquement les noms d'hôtes partiels en l'absence de proxy explicite.

Voici ce qui se passe dans cette figure :

• À l'étape 1, l'utilisateur saisit « oreilly » dans la fenêtre URI du navigateur. Le navigateur utilise « oreilly » comme nom d'hôte et utilise

comme schéma par défaut « http:// », port par défaut « 80 » et chemin par défaut « / ».

• À l'étape 2a, le navigateur recherche l'hôte « oreilly ». Cela échoue.

* La plupart des navigateurs vous permettent de saisir « yahoo » et de le développer automatiquement en « www.yahoo.com ». De même, certains navigateurs vous permettent d'omettre le préfixe « http:// » et de l'insérer s'il est manquant.

• À l'étape 3a, le navigateur étend automatiquement le nom d'hôte et demande au DNS de résoudre

« www.oreilly.com ». C'est réussi.

• Le navigateur se connecte ensuite avec succès à www.oreilly.com.

Résolution d'URI avec un proxy explicite

Lorsque vous utilisez un proxy explicite, le navigateur n'effectue plus aucune de ces extensions de commodité, car l'URI de l'utilisateur est transmis directement au proxy.

Comme le montre la figure 6-17, le navigateur ne développe pas automatiquement le nom d'hôte partiel lorsqu'un proxy explicite est présent. Par conséquent, lorsque l'utilisateur saisit « oreilly » dans la fenêtre d'emplacement du navigateur, le proxy reçoit « http://oreilly/ » (le navigateur ajoute le schéma et le chemin par défaut, mais conserve le nom d'hôte tel qu'il a été saisi).

Figure 6-17. Le navigateur ne développe pas automatiquement les noms d'hôtes partiels en présence d'un proxy explicite.

Pour cette raison, certains proxys tentent d'imiter autant que possible les services pratiques du navigateur, y compris l'extension automatique « www...com » et l'ajout de suffixes de domaine local.*

Résolution d'URI avec un proxy d'interception

La résolution des noms d'hôtes est légèrement différente avec un proxy d'interception invisible, car pour le client, il n'y a pas de proxy ! Le comportement est similaire à celui du serveur : le navigateur étend automatiquement les noms d'hôtes jusqu'à la réussite du DNS. Cependant, une différence significative se produit lors de la connexion au serveur, car

La figure 6-18 l'illustre.

* Mais, pour les proxys largement partagés, il peut être impossible de connaître le suffixe de domaine approprié pour les utilisateurs individuels.

Figure 6-18. Le navigateur ne détecte pas les adresses IP des serveurs inactifs lors de l'utilisation de proxys d'interception.

La figure 6-18 illustre la transaction suivante :

• À l'étape 1, l'utilisateur tape « oreilly » dans la fenêtre d'emplacement de l'URI du navigateur.

• À l'étape 2a, le navigateur recherche l'hôte « oreilly » via DNS, mais le serveur DNS échoue et répond que l'hôte est inconnu, comme indiqué à l'étape 2b.

• À l'étape 3a, le navigateur effectue une extension automatique, convertissant « oreilly » en « www.oreilly.com ». À l'étape 3b, le navigateur recherche l'hôte « www.oreilly.com » via DNS. Cette fois, comme illustré à l'étape 3c, le serveur DNS réussit et renvoie les adresses IP au navigateur.

• À l'étape 4a, le client a déjà résolu le nom d'hôte et dispose d'une liste d'adresses IP. Normalement, le client tente de se connecter à chaque adresse IP jusqu'à ce qu'il y parvienne, car certaines adresses IP peuvent être inactives. Mais avec un proxy intercepteur, la première tentative de connexion est interrompue par le serveur proxy, et non par le serveur d'origine. Le client pense communiquer avec le serveur web, mais celui-ci n'est peut-être même pas actif.

• Lorsque le proxy est enfin prêt à interagir avec le serveur d'origine réel (étape 5b), il peut constater que l'adresse IP pointe en réalité vers un serveur hors service. Pour offrir le même niveau de tolérance aux pannes que le navigateur, le proxy doit essayer d'autres adresses IP, soit en résolvant le nom d'hôte dans l'en-tête Host, soit en effectuant une recherche DNS inversée sur l'adresse IP. Il est important que les implémentations de proxy d'interception et de proxy explicite prennent en charge la tolérance aux pannes lors de la résolution DNS des serveurs hors service, car lorsque les navigateurs sont configurés pour utiliser un proxy explicite, ils s'appuient sur ce proxy pour sa tolérance aux pannes.

Suivi des messages

Aujourd'hui, il n'est pas rare que les requêtes web transitent par une chaîne de deux ou plusieurs proxys entre le client et le serveur (Figure 6-19). Par exemple, de nombreuses entreprises utilisent des serveurs proxy de mise en cache pour accéder à Internet, pour des raisons de sécurité et de réduction des coûts, et de nombreux grands FAI utilisent des caches proxy pour améliorer les performances et implémenter des fonctionnalités. Un pourcentage important des requêtes web transitent aujourd'hui par des proxys. Parallèlement, la réplication de contenu sur des banques de caches de substitution réparties dans le monde entier devient de plus en plus courante, pour des raisons de performances.

Figure 6-19. Les proxys d'accès et les proxys CDN créent des hiérarchies de proxy à deux niveaux.

Les proxys sont développés par différents fournisseurs. Ils présentent des fonctionnalités et des bugs différents et sont administrés par différentes organisations.

À mesure que les proxys deviennent plus répandus, vous devez être en mesure de suivre le flux de messages entre les proxys et de détecter tout problème, tout comme il est important de suivre le flux de paquets IP entre différents commutateurs et routeurs.

L'en-tête Via

Le champ d'en-tête Via répertorie les informations relatives à chaque nœud intermédiaire (proxy ou passerelle) traversé par un message. Chaque fois qu'un message transite par un autre nœud, ce dernier doit être ajouté à la fin de la liste Via.

La chaîne Via suivante indique que le message a transité par deux proxys. Elle indique que le premier proxy implémentait le protocole HTTP/1.1 et s'appelait proxy-62.irenes-isp.net, et que le second implémentait HTTP/1.0 et s'appelait cache.joes-hardware.com :

Via : 1.1 proxy-62.irenes-isp.net, 1.0 cache.joes-hardware.com

Le champ d'en-tête Via est utilisé pour suivre la transmission des messages, diagnostiquer les boucles de messages et identifier les capacités de protocole de tous les expéditeurs le long de la chaîne de demande/réponse (Figure 6-20).

Les proxys peuvent également utiliser les en-têtes Via pour détecter les boucles de routage sur le réseau. Un proxy doit insérer une chaîne unique associée à lui-même dans l'en-tête Via avant d'envoyer une requête et vérifier la présence de cette chaîne dans les requêtes entrantes afin de détecter les boucles de routage sur le réseau.

Figure 6-20. Exemple d'en-tête Via

Via la syntaxe

Le champ d'en-tête « Via » contient une liste de points de cheminement séparés par des virgules. Chaque point de cheminement représente un serveur proxy ou une passerelle et contient des informations sur le protocole et l'adresse de ce nœud intermédiaire. Voici un exemple d'en-tête « Via » avec deux points de cheminement :

Via = 1.1 cache.joes-hardware.com, 1.1 proxy.irenes-isp.net La syntaxe formelle d'un en-tête Via est présentée ici :

Via = "Via" ":" 1#(point de cheminement)

waypoint = (protocole-reçu reçu par [commentaire]) protocole-reçu = [nom-protocole "/"] version-protocole reçu par = (hôte [":" port]) | pseudonyme

Notez que chaque point de cheminement Via contient jusqu'à quatre composants : un nom de protocole facultatif (par défaut HTTP), une version de protocole requise, un nom de nœud requis et un commentaire descriptif facultatif :

Nom du protocole

Protocole reçu par un intermédiaire. Le nom du protocole est facultatif s'il s'agit de HTTP. Sinon, le nom du protocole est ajouté à la version, séparé par un « / ». Des protocoles non HTTP peuvent être utilisés lors de la connexion de passerelles.

Requêtes HTTP pour d'autres protocoles (HTTPS, FTP, etc.).

Version du protocole

Version du message reçu. Son format dépend du protocole. Pour HTTP, les numéros de version standard sont utilisés (« 1.0 », « 1.1 », etc.). La version est indiquée dans le champ « Via », afin que les applications ultérieures connaissent les capacités du protocole de tous les intermédiaires précédents.

Nom du nœud

L'hôte et le numéro de port facultatif de l'intermédiaire (si le port n'est pas indiqué, vous pouvez utiliser le port par défaut du protocole). Dans certains cas, une organisation peut ne pas vouloir divulguer le véritable nom d'hôte, pour des raisons de confidentialité ; dans ce cas, il peut être remplacé par un pseudonyme.

Commentaire du nœud

Un commentaire facultatif décrivant plus en détail le nœud intermédiaire. Il est courant d'inclure ici des informations sur le fournisseur et la version, et certains serveurs proxy utilisent également le champ de commentaire pour inclure des informations de diagnostic sur les événements survenus sur ce périphérique.*

Via les chemins de requête et de réponse

Les messages de demande et de réponse passent tous deux par des proxys, de sorte que les messages de demande et de réponse ont tous deux des en-têtes Via.

Étant donné que les requêtes et les réponses transitent généralement par la même connexion TCP, les messages de réponse empruntent le même chemin que les requêtes. Si une requête transite par les proxys A, B et C, le message de réponse correspondant transite par les proxys C, B, puis A. Ainsi, l'en-tête Via des réponses est presque toujours l'inverse de l'en-tête Via des réponses (Figure 6-21).

Figure 6-21. La réponse Via est généralement l'inverse de la requête Via.

Via et passerelles

Certains proxys offrent une fonctionnalité de passerelle aux serveurs utilisant des protocoles non HTTP. L'en-tête Via enregistre ces conversions de protocole, permettant ainsi aux applications HTTP de connaître les capacités et les conversions de protocole tout au long de la chaîne de proxy. La figure 6-22 illustre un client HTTP demandant une URI FTP via une passerelle HTTP/FTP.

Le client envoie une requête HTTP pour ftp://http-guide.com/pub/welcome.txt à la passerelle proxy.irenes-isp.net. Le proxy, agissant comme passerelle de protocole, récupère l'objet souhaité auprès du serveur FTP, via le protocole FTP. Il renvoie ensuite l'objet au client dans une réponse HTTP, avec le champ d'en-tête Via suivant :

Via : FTP/1.0 proxy.irenes-isp.net (Traffic-Server/5.0.1-17882 [cMs f])

* Par exemple, les serveurs proxy de mise en cache peuvent inclure des informations de succès/échec.

Figure 6-22. La passerelle HTTP/FTP génère des en-têtes Via, enregistrant le protocole reçu (FTP)

Notez que le protocole reçu est FTP. Le commentaire facultatif contient la marque et le numéro de version du serveur proxy, ainsi que des informations de diagnostic du fournisseur. Vous trouverez plus d'informations sur les passerelles au chapitre 8.

Les en-têtes Server et Via

L'en-tête de réponse du serveur décrit le logiciel utilisé par le serveur d'origine. Voici quelques exemples :

Serveur : Apache/1.3.14 (Unix) PHP/4.0.4

Serveur : Netscape-Enterprise/4.1

Serveur : Microsoft-IIS/5.0

Si un message de réponse est transmis via un proxy, assurez-vous que celui-ci ne modifie pas l'en-tête du serveur. Cet en-tête est destiné au serveur d'origine.

Au lieu de cela, le proxy doit ajouter une entrée Via.

Conséquences de Via sur la confidentialité et la sécurité

Dans certains cas, nous ne souhaitons pas inclure les noms d'hôtes exacts dans la chaîne Via. En général, sauf si ce comportement est explicitement activé, lorsqu'un serveur proxy fait partie d'un pare-feu réseau, il ne doit pas transmettre les noms et ports des hôtes situés derrière le pare-feu, car la connaissance de l'architecture réseau derrière un pare-feu peut être utile.

partie malveillante.*

* Les personnes malveillantes peuvent utiliser les noms et les numéros de version des ordinateurs pour en savoir plus sur l'architecture réseau derrière un périmètre de sécurité. Ces informations peuvent être utiles lors d'attaques. De plus, les noms des ordinateurs peuvent révéler des indices sur des projets privés au sein d'une organisation.

Si la redirection de nom de nœud Via n'est pas activée, les proxys faisant partie d'un périmètre de sécurité doivent remplacer le nom d'hôte par un pseudonyme approprié. Cependant, en règle générale, les proxys doivent s'efforcer de conserver une entrée de point de cheminement Via pour chaque serveur proxy, même si le nom réel est masqué.

Pour les organisations ayant des exigences de confidentialité très strictes pour masquer la conception et la topologie de leurs architectures réseau internes, un proxy peut combiner une séquence ordonnée d'entrées de point de cheminement Via (avec des valeurs de protocole de réception identiques) en une seule entrée jointe. Par exemple :

Via : 1.0 foo, 1.1 devirus.company.com, 1.1 access-logger.company.com pourraient être réduits à :

Via : 1.0 foo, 1.1 hidden-stuff

Ne combinez pas plusieurs entrées, sauf si elles relèvent toutes du même contrôle organisationnel et que les hôtes ont déjà été remplacés par des pseudonymes. De même, ne combinez pas des entrées ayant des valeurs de protocole de réception différentes.

La méthode TRACE

Les serveurs proxy peuvent modifier les messages au fur et à mesure de leur transmission. Les en-têtes sont ajoutés, modifiés et supprimés, et le corps du message peut être converti en différents formats. À mesure que les proxys deviennent plus sophistiqués et que de plus en plus de fournisseurs déploient des produits proxy, les problèmes d'interopérabilité augmentent. Pour diagnostiquer facilement les réseaux proxy, nous devons pouvoir observer l'évolution des messages lors de leur transmission, étape par étape, via le réseau proxy HTTP.

La méthode TRACE de HTTP/1.1 permet de tracer un message de requête à travers une chaîne de proxys, en observant par quels proxys

le message transite et comment chaque proxy modifie le message. TRACE est très utile pour déboguer les flux proxy.*

Lorsque la requête TRACE atteint le serveur de destination†, l'intégralité du message est renvoyée à l'expéditeur, regroupée dans le corps d'une réponse HTTP (voir Figure 6-23). À l'arrivée de la réponse TRACE, le client peut examiner le message exact reçu par le serveur et la liste des proxys par lesquels il est passé (dans l'en-tête Via). La réponse TRACE a le type de contenu « message/http » et une erreur 200.

Statut OK.

Max-Forwards

Normalement, les messages TRACE parviennent jusqu'au serveur de destination, quel que soit le nombre de proxys intermédiaires. Vous pouvez utiliser l'en-tête Max-Forwards pour limiter le nombre de messages.

* Malheureusement, ce n’est pas encore largement mis en œuvre.

† Le destinataire final est soit le serveur d'origine, soit le premier proxy ou passerelle à recevoir une valeur Max-Forwards de zéro (0) dans la demande.

Figure 6-23. La réponse TRACE reflète le message de requête reçu.

Le nombre de sauts de proxy pour les requêtes TRACE et OPTIONS, utile pour tester une chaîne de proxys transférant des messages en boucle infinie ou pour vérifier l'effet de serveurs proxy particuliers au milieu d'une chaîne. Max-Forwards limite également la transmission des messages OPTIONS (voir « Interopérabilité des proxys »).

L'en-tête de requête Max-Forwards contient un entier unique indiquant le nombre de fois restantes que ce message peut être transmis (Figure 6-24). Si la valeur Max-Forwards est nulle (Max-Forwards : 0), le destinataire doit renvoyer le message TRACE au client sans le transmettre, même s'il n'est pas le serveur d'origine.

Si la valeur Max-Forwards reçue est supérieure à zéro, le message transféré doit contenir un champ Max-Forwards mis à jour avec une valeur décrémentée de 1. Tous les proxys et passerelles doivent prendre en charge Max-Forwards. Vous pouvez utiliser Max-Forwards pour visualiser la requête à n'importe quel saut d'une chaîne de proxy.

Authentification proxy

Les proxys peuvent servir de dispositifs de contrôle d'accès. HTTP définit un mécanisme appelé authentification proxy qui bloque les requêtes de contenu jusqu'à ce que l'utilisateur fournisse des informations d'identification valides au proxy :

• Lorsqu'une demande de contenu restreint arrive sur un serveur proxy, le serveur proxy peut renvoyer un code d'état 407 Autorisation de proxy requise demandant l'accès

Figure 6-24. Vous pouvez limiter le nombre de sauts de transfert avec le champ d'en-tête Max-Forwards.

informations d'identification, accompagnées d'un champ d'en-tête Proxy-Authenticate qui décrit comment fournir ces informations d'identification (Figure 6-25b).

• Lorsque le client reçoit la réponse 407, il tente de collecter les informations d'identification requises, soit à partir d'une base de données locale, soit en demandant à l'utilisateur.

• Une fois les informations d'identification obtenues, le client renvoie la demande en fournissant les informations d'identification requises dans un champ d'en-tête Proxy-Authorization.

• Si les informations d'identification sont valides, le proxy transmet la demande d'origine le long de la chaîne (Figure 6-25c) ; sinon, une autre réponse 407 est envoyée.

L'authentification par proxy ne fonctionne généralement pas correctement lorsqu'une chaîne comporte plusieurs proxys, chacun participant à l'authentification. Des améliorations ont été proposées à HTTP pour associer les informations d'authentification à des points de passage spécifiques dans une chaîne de proxy, mais ces améliorations n'ont pas été largement mises en œuvre.

Assurez-vous de lire le chapitre 12 pour une explication détaillée des mécanismes d'authentification HTTP.

Interopérabilité des proxys

Les clients, serveurs et proxys sont développés par plusieurs fournisseurs, selon différentes versions de la spécification HTTP. Ils prennent en charge diverses fonctionnalités et présentent des bugs variés. Les serveurs proxy doivent servir d'intermédiaire entre les périphériques côté client et côté serveur, qui peuvent implémenter des protocoles différents et présenter des problèmes.

Interopérabilité des proxys

Figure 6-25. Les proxys peuvent implémenter l'authentification pour contrôler l'accès au contenu.

Gestion des en-têtes et des méthodes non pris en charge

Il est possible que le serveur proxy ne comprenne pas tous les champs d'en-tête qui le traversent. Certains en-têtes peuvent être plus récents que le proxy lui-même ; d'autres peuvent être des champs d'en-tête personnalisés, propres à une application particulière. Les proxys doivent transmettre les champs d'en-tête non reconnus et conserver l'ordre relatif des champs d'en-tête portant le même nom.* De même, si un proxy ne maîtrise pas une méthode, il doit, si possible, tenter de transmettre le message au saut suivant.

Les proxys qui ne peuvent pas tunneliser des méthodes non prises en charge peuvent ne pas être viables dans la plupart des réseaux actuels, car l'accès à Hotmail via Microsoft Outlook utilise largement les méthodes d'extension HTTP.

* Un message peut contenir plusieurs champs d'en-tête portant le même nom. Dans ce cas, ils doivent pouvoir être combinés de manière équivalente dans une liste séparée par des virgules. L'ordre de réception des champs d'en-tête portant le même nom est donc déterminant pour l'interprétation de la valeur du champ combiné. Un proxy ne peut donc pas modifier l'ordre relatif de ces valeurs de champs portant le même nom lors de la transmission d'un message.

OPTIONS : Découverte de la prise en charge des fonctionnalités facultatives

La méthode HTTP OPTIONS permet à un client (ou proxy) de découvrir les fonctionnalités prises en charge (par exemple, les méthodes prises en charge) d'un serveur web ou d'une ressource spécifique sur un serveur web (Figure 6-26). Les clients peuvent utiliser OPTIONS pour déterminer les capacités d'un serveur avant d'interagir avec celui-ci, ce qui facilite l'interopérabilité avec des proxys et des serveurs de différents niveaux de fonctionnalités.

Figure 6-26. Utilisation d'OPTIONS pour trouver les méthodes prises en charge par un serveur

Si l'URI de la requête OPTIONS est un astérisque (*), la requête concerne l'ensemble des fonctionnalités prises en charge par le serveur. Par exemple :

OPTIONS * HTTP/1.1

Si l'URI est une ressource réelle, la requête OPTIONS s'enquiert des fonctionnalités disponibles pour cette ressource particulière :

OPTIONS http://www.joes-hardware.com/index.html HTTP/1.1

En cas de succès, la méthode OPTIONS renvoie une réponse 200 OK incluant divers champs d'en-tête décrivant les fonctionnalités facultatives prises en charge par le serveur ou disponibles pour la ressource. Le seul champ d'en-tête spécifié par HTTP/1.1 dans la

réponse est l'en-tête Allow, qui décrit les méthodes prises en charge par le serveur (ou une ressource spécifique sur le serveur).* OPTIONS autorise un corps de réponse facultatif contenant davantage d'informations, mais ce champ n'est pas défini.

L'en-tête Allow

Le champ d'en-tête d'entité « Autoriser » répertorie l'ensemble des méthodes prises en charge par la ressource identifiée par l'URI de la requête, ou par l'ensemble du serveur si l'URI de la requête est *. Par exemple :

Autoriser : GET, HEAD, PUT

L'en-tête Allow peut être utilisé comme en-tête de requête pour recommander les méthodes prises en charge par la nouvelle ressource. Le serveur n'est pas tenu de prendre en charge ces méthodes.

* Toutes les ressources ne prennent pas en charge toutes les méthodes. Par exemple, une requête de script CGI peut ne pas prendre en charge une méthode PUT, et un fichier HTML statique n'acceptera pas une méthode POST.

Interopérabilité des proxys

et doit inclure un en-tête Allow dans la réponse correspondante, répertoriant les méthodes réellement prises en charge.

Un proxy ne peut pas modifier le champ d'en-tête Autoriser même s'il ne comprend pas toutes les méthodes spécifiées, car le client peut avoir d'autres chemins pour communiquer avec le serveur d'origine.

Pour plus d'informations

Pour plus d'informations, reportez-vous à :

http://www.w3.org/Protocols/rfc2616/rfc2616.txt

RFC 2616, « Protocole de transfert hypertexte », par R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Mastinter, P. Leach et T. Berners-Lee.

http://search.ietf.org/rfc/rfc3040.txt

RFC 3040, « Taxonomie de la réplication et de la mise en cache du Web Internet ».

Serveurs proxy Web

Ari Luotonen, Livres informatiques de Prentice Hall.

http://search.ietf.org/rfc/rfc3143.txt

RFC 3143, « Problèmes connus de proxy/mise en cache HTTP ».

Mise en cache Web

Duane Wessels, O'Reilly & Associates, Inc.

Chapitre 7Ceci est le titre du livre CHAPITRE 7

Mise en cache

Les caches Web sont des dispositifs HTTP qui conservent automatiquement des copies des documents populaires. Lorsqu'une requête Web parvient à un cache, si une copie locale « mise en cache » est disponible, le document est servi depuis le stockage local plutôt que depuis le serveur d'origine.

Les caches présentent les avantages suivants :

• Les caches réduisent les transferts de données redondants, vous permettant ainsi d'économiser de l'argent sur les frais de réseau.

• Les caches réduisent les goulots d'étranglement du réseau. Les pages se chargent plus rapidement sans augmentation de la bande passante.

• Les caches réduisent la demande sur les serveurs d'origine. Les serveurs répondent plus rapidement et évitent la surcharge.

• Les caches réduisent les délais de distance, car les pages se chargent plus lentement à plus grande distance.

Dans ce chapitre, nous expliquons comment les caches améliorent les performances et réduisent les coûts, comment mesurer leur efficacité et où les placer pour maximiser leur impact. Nous expliquons également comment HTTP maintient les copies en cache à jour et comment les caches interagissent avec d'autres caches et serveurs.

Transferts de données redondants

Lorsque plusieurs clients accèdent à une page d'un serveur d'origine populaire, le serveur transmet le même document plusieurs fois, une fois à chaque client. Les mêmes octets circulent indéfiniment sur le réseau. Ces transferts de données redondants consomment une bande passante réseau coûteuse, ralentissent les transferts et surchargent les serveurs web. Grâce aux caches, le cache conserve une copie de la première réponse du serveur. Les requêtes suivantes peuvent être traitées à partir de cette copie mise en cache, réduisant ainsi le trafic inutile et dupliqué vers et depuis les serveurs d'origine.

Goulots d'étranglement de la bande passante

Les caches peuvent également réduire les goulots d'étranglement du réseau. De nombreux réseaux offrent davantage de bande passante aux clients locaux qu'aux serveurs distants (Figure 7-1). Les clients accèdent aux serveurs à la vitesse du réseau le plus lent. Si un client obtient une copie d'un cache sur un réseau local rapide, la mise en

cache peut améliorer les performances, notamment pour les documents volumineux.

Figure 7-1. La bande passante limitée sur une zone étendue crée un goulot d'étranglement que les caches peuvent améliorer.

Dans la figure 7-1, il peut falloir 30 secondes à un utilisateur de la succursale de Joe's Hardware, Inc. à San Francisco pour télécharger un fichier d'inventaire de 5 Mo depuis le siège d'Atlanta, via la connexion Internet T1 à 1,4 Mbit/s. Si le document était mis en cache dans le bureau de San Francisco, un utilisateur local pourrait obtenir le même document en moins d'une seconde via la connexion Ethernet.

Le tableau 7-1 montre l'impact de la bande passante sur le temps de transfert en fonction de la vitesse du réseau et de la taille des documents. La bande passante entraîne des retards notables pour les documents volumineux, et la différence de vitesse entre les différents types de réseau est considérable.* Un modem à 56 Kbit/s prendrait 749 secondes (plus de 12 minutes) pour transférer un

Fichier de 5 Mo pouvant être transporté en moins d'une seconde sur un réseau local Ethernet rapide.

Tableau 7-1. Délais de transfert imposés par la bande passante, idéalisés (temps en secondes)

Grand HTML (15 Ko) JPEG (40 Ko) Grand JPEG (150 Ko) Grand fichier (5 Mo)

Modem commuté (56 Kbit/sec) 2,19 5,85 21,94 748,98

DSL (256 Kbit/s) 0,48 1,28 4,80 163,84

T1 (1,4 Mbit/s) 0,09 0,23 0,85 29,13

Ethernet lent (10 Mbit/s) .01 .03 .12 4.19

DS3 (45 Mbit/s) .00 .01 .03 .93

Ethernet rapide (100 Mbit/s) .00 .00 .01 .42

* Ce tableau montre uniquement l'effet de la bande passante réseau sur le temps de transfert. Il suppose une efficacité réseau de 100 % et l'absence de latences de traitement réseau ou applicative. De ce fait, le délai est une borne inférieure. Les délais réels seront plus importants, et ceux des petits objets seront dominés par des surcharges non liées à la bande passante.

Foules éclair

La mise en cache est particulièrement importante pour disperser les foules soudaines. Ces foules se produisent lorsqu'un événement

soudain (comme une actualité de dernière minute, une annonce massive par courriel ou un événement mettant en scène une célébrité) incite de nombreuses personnes à accéder à un document web quasiment simultanément (Figure 7-2). Le pic de trafic redondant qui en résulte peut provoquer une panne catastrophique des réseaux et des serveurs web.

Figure 7-2. Les foules soudaines peuvent surcharger les serveurs web

Lorsque le « rapport Starr », détaillant l'enquête de Kenneth Starr sur le président américain Clinton, a été publié sur Internet le 11 septembre 1998, les serveurs web de la Chambre des représentants des États-Unis ont reçu plus de 3 millions de requêtes par heure, soit 50 fois la charge moyenne d'un serveur. Un site d'information, CNN.com, a signalé une moyenne de plus de 50 000 requêtes par seconde sur ses serveurs.

Retards de distance

Même si la bande passante n'est pas un problème, la distance peut l'être. Chaque routeur réseau ajoute des délais au trafic Internet. Et même s'il n'y a pas beaucoup de routeurs entre le client et le serveur, la vitesse de la lumière à elle seule peut entraîner un délai important.

La distance directe entre Boston et San Francisco est d'environ 4 300 kilomètres. Dans le meilleur des cas, à la vitesse de la lumière (300 000 kilomètres par seconde), un signal pourrait voyager de Boston à San Francisco en environ 15 millisecondes et effectuer un aller-retour en 30 millisecondes*.

* En réalité, les signaux voyagent à une vitesse légèrement inférieure à celle de la lumière, donc les retards de distance sont encore pires.

Retards de distance

Imaginons qu'une page web contienne 20 petites images, toutes hébergées sur un serveur à San Francisco. Si un client à Boston ouvre quatre connexions parallèles au serveur et les maintient actives, la vitesse de la lumière à elle seule contribue à près d'un quart de seconde (240 ms) au temps de téléchargement (Figure 7-3). Si le serveur est à Tokyo (à 10 700 km de Boston), le délai atteint 600 ms. Les pages web moyennement complexes peuvent subir plusieurs secondes de retard à la vitesse de la lumière.

Figure 7-3. La vitesse de la lumière peut entraîner des retards importants, même avec des connexions parallèles et permanentes.

Placer des caches dans les salles de machines à proximité peut réduire la distance de déplacement des documents de milliers de kilomètres à quelques dizaines de mètres.

Succès et échecs

Les caches peuvent donc être utiles. Mais ils ne stockent pas une copie de tous les documents du monde.*

* Rares sont ceux qui peuvent se permettre d'acheter une mémoire cache suffisamment grande pour contenir tous les documents du Web. Et même si l'on pouvait se permettre d'avoir des « mémoires Web complètes » gigantesques, certains documents changent si fréquemment qu'ils ne seront pas toujours d'actualité dans de nombreuses mémoires cache.

Certaines requêtes arrivant dans un cache peuvent être traitées à partir d'une copie disponible. On parle alors d'un succès de cache (Figure 7-4a). D'autres requêtes arrivent dans un cache et sont transmises au serveur d'origine, faute de copie disponible. On parle alors d'un échec de cache (Figure 7-4b).

Figure 7-4. Succès, échecs et revalidations du cache

Revalidations

Le contenu du serveur d'origine étant susceptible de changer, les caches doivent vérifier régulièrement que leurs copies sont toujours à jour avec le serveur. Ces « vérifications de fraîcheur » sont appelées revalidations HTTP (Figure 7-4c). Pour optimiser ces revalidations, HTTP définit des requêtes spécifiques permettant de vérifier rapidement si le contenu est toujours à jour, sans avoir à récupérer l'objet entier depuis le serveur.

Un cache peut revalider une copie à tout moment et aussi souvent qu'il le souhaite. Cependant, comme les caches contiennent souvent des millions de documents et que la bande passante réseau est limitée, la plupart des caches ne revalident une copie que lorsqu'elle est demandée par un client et que la copie est suffisamment ancienne pour justifier une vérification. Nous expliquerons les règles HTTP de vérification de l'actualité plus loin dans ce chapitre.

Lorsqu'un cache doit revalider une copie mise en cache, il envoie une courte requête de revalidation au serveur d'origine. Si le contenu n'a pas changé, le serveur renvoie une brève réponse 304 « Non modifié ». Dès que le cache constate que la copie est toujours valide, il la marque à nouveau comme étant temporairement actualisée et la transmet au client (Figure 7-5a). On parle alors de « revalidate hit » ou de « slow hit ». Ce résultat est plus lent qu'un simple hit de cache, car il nécessite

une vérification auprès du serveur d'origine, mais plus rapide qu'un « cache miss », car aucune donnée d'objet n'est récupérée du serveur.

Succès et échecs

Figure 7-5. Les revalidations réussies sont plus rapides que les échecs de cache ; les revalidations échouées sont presque identiques aux échecs.

HTTP propose plusieurs outils pour revalider les objets mis en cache, mais le plus populaire est l'en-tête If-Modified-Since. Ajouté à une requête GET, cet en-tête indique au serveur de n'envoyer l'objet que s'il a été modifié depuis sa mise en cache.

Voici ce qui se passe lorsqu'une requête GET If-Modified-Since arrive au serveur dans trois circonstances : lorsque le contenu du serveur n'est pas modifié, lorsque le contenu du serveur a été modifié et lorsque l'objet sur le serveur est supprimé :

Revalider le hit

Si l'objet serveur n'est pas modifié, le serveur envoie au client une courte réponse HTTP 304 « Non modifié ». Ceci est illustré à la figure 7-6.

Figure 7-6. HTTP utilise l'en-tête If-Modified-Since pour la revalidation

Revalider l'échec

Si l'objet serveur est différent de la copie mise en cache, le serveur envoie au client une réponse HTTP 200 OK normale, avec le contenu complet.

Objet supprimé

Si l'objet serveur a été supprimé, le serveur renvoie une réponse 404 Not Found et le cache supprime sa copie.

Taux de réussite

La fraction des requêtes traitées depuis le cache est appelée taux de réussite du cache (ou taux de réussite du cache)*, ou parfois taux de réussite des documents (ou taux de réussite des documents). Ce taux varie de 0 à 1, mais est souvent exprimé en pourcentage : 0 % signifie que chaque requête a échoué (le document a dû être transmis sur le réseau) et 100 %.

signifie que chaque requête était un succès (avait une copie dans le cache).†

Les administrateurs de cache souhaitent un taux de réussite proche de 100 %. Le taux de réussite réel dépend de la taille de votre cache, de la

similitude des centres d'intérêt des utilisateurs, de la fréquence de modification ou de personnalisation des données mises en cache, et de la configuration des caches. Le taux de réussite est notoirement difficile à prévoir, mais un taux de réussite de 40 % est acceptable pour un cache web modeste aujourd'hui. L'avantage des caches est que même un cache de taille modeste peut contenir suffisamment de documents populaires pour améliorer considérablement les performances et réduire le trafic. Les caches s'efforcent de conserver le contenu utile.

Taux de réussite des octets

Le taux de réussite des documents n'est toutefois pas exhaustif, car ils n'ont pas tous la même taille. Certains objets volumineux peuvent être consultés moins souvent, mais contribuent davantage au trafic global de données, en raison de leur taille. C'est pourquoi certains préfèrent la mesure du taux de réussite en octets (surtout ceux qui sont facturés à l'octet de trafic !).

Le taux de réussite représente la fraction de tous les octets transférés qui ont été servis depuis le cache. Cette mesure reflète le degré d'économie de trafic. Un taux de réussite de 100 % signifie que chaque octet provient du cache et qu'aucun trafic n'est passé sur Internet.

Le taux de réussite des documents et le taux de réussite des octets sont deux indicateurs utiles des performances du cache. Le taux de réussite des documents décrit le nombre de transactions web exclues du réseau sortant. Les transactions ayant une composante temporelle fixe et souvent importante (par exemple, l'établissement d'une connexion TCP à un serveur), l'amélioration du taux de réussite des documents optimisera la réduction globale de la latence (délai). Le taux de réussite des octets décrit le nombre d'octets exclus d'Internet. L'amélioration de ce taux optimisera les économies de bande passante.

* Le terme « taux de réussite » est probablement plus approprié que « taux de réussite », car ce terme évoque à tort un facteur temps. Cependant, ce terme est d'usage courant, c'est pourquoi nous l'utilisons ici.

† Il arrive que les revalidations soient prises en compte dans le taux de réussite, mais il arrive aussi que ces deux taux soient mesurés séparément. Lorsque vous analysez les taux de réussite, assurez-vous de bien comprendre ce qui constitue une « réussite ».

Succès et échecs

Distinguer les succès et les échecs

Malheureusement, HTTP ne permet pas au client de savoir si une réponse est un accès au cache ou un accès au serveur d'origine. Dans les deux cas, le code de réponse sera 200 OK, indiquant que la réponse contient un corps. Certains caches proxy commerciaux ajoutent des

informations supplémentaires aux en-têtes Via pour décrire ce qui s'est passé dans le cache.

Un client peut généralement détecter si la réponse provient d'un cache en utilisant l'en-tête Date. En comparant la valeur de l'en-tête Date de la réponse à l'heure actuelle, un client peut souvent détecter une réponse en cache grâce à sa date la plus ancienne. Un autre moyen de détecter une réponse en cache est l'en-tête Age, qui indique l'ancienneté de la réponse (voir « Âge » à l'annexe C).

Topologies de cache

Les caches peuvent être dédiés à un seul utilisateur ou partagés entre des milliers d'utilisateurs. Les caches dédiés sont appelés caches privés. Les caches privés sont des caches personnels, contenant les pages populaires pour un seul utilisateur (Figure 7-7a). Les caches partagés sont appelés caches publics. Les caches publics contiennent les pages populaires de la communauté d'utilisateurs (Figure 7-7b).

Figure 7-7. Caches publiques et privées

Caches privées

Les caches privés ne nécessitent pas beaucoup de puissance ni d'espace de stockage, ce qui permet de les rendre compacts et économiques. Les navigateurs web intègrent des caches privés : la plupart des navigateurs mettent en cache les documents les plus courants sur le disque et la mémoire de votre ordinateur et vous permettent de configurer la taille et les paramètres du cache. Vous pouvez également consulter le contenu des caches des navigateurs. Par exemple, avec Microsoft Internet Explorer, vous pouvez accéder au contenu du cache depuis la boîte de dialogue Outils → Options Internet…. MSIE appelle les documents mis en cache « Fichiers temporaires » et les répertorie dans un fichier, avec les URL associées et les dates d'expiration des documents. Vous pouvez consulter le contenu du cache de Netscape Navigator via l'URL spéciale about:cache, qui affiche une page « Statistiques du cache disque » affichant le contenu du cache.

Caches proxy publics

Les caches publics sont des serveurs proxy spécifiques et partagés, appelés serveurs proxy de mise en cache ou, plus communément, caches proxy (les proxys ont été abordés au chapitre 6). Les caches proxy traitent les documents du cache local ou contactent le serveur pour le compte de l'utilisateur. Étant donné qu'un cache public reçoit des accès de plusieurs utilisateurs, il a plus de chances d'éliminer le trafic redondant.*

Dans la figure 7-8a, chaque client accède de manière redondante à un nouveau document « chaud » (pas encore dans le cache privé). Chaque cache privé récupère le même document, traversant le réseau plusieurs fois. Avec un cache public partagé, comme dans la figure 7-8b, le cache n'a besoin de récupérer l'objet populaire qu'une seule fois et utilise la copie partagée pour traiter toutes les requêtes, réduisant ainsi le trafic réseau.

Les caches proxy suivent les règles relatives aux proxys décrites au chapitre 6. Vous pouvez configurer votre navigateur pour utiliser un cache proxy en spécifiant un proxy manuel ou en configurant un fichier de configuration automatique de proxy (voir « Configuration manuelle du proxy client » au chapitre 6). Vous pouvez également forcer les requêtes HTTP à passer par les caches sans configurer votre navigateur en utilisant des proxys d'interception (voir chapitre 20).

Hiérarchies de cache proxy

En pratique, il est souvent judicieux de déployer des hiérarchies de caches, où les échecs de cache dans les caches plus petits sont canalisés vers des caches parents plus grands qui traitent le trafic « distillé » restant. La figure 7-9 illustre une hiérarchie de cache à deux niveaux†. L'idée est d'utiliser des caches petits et peu coûteux à proximité des clients, puis des caches progressivement plus grands et plus puissants vers le haut de la hiérarchie pour stocker les documents partagés par de nombreux utilisateurs‡.

Il est à espérer que la plupart des utilisateurs obtiendront des réponses positives sur les caches de niveau 1 proches (comme illustré à la figure 7-9a). Dans le cas contraire, des caches parents plus volumineux pourront traiter leurs requêtes (figure 7-9b). Pour les hiérarchies de cache profondes, il est possible de parcourir de longues chaînes de

* Étant donné qu'un cache public met en cache les divers intérêts de la communauté des utilisateurs, il doit être suffisamment grand pour contenir un ensemble de documents populaires, sans être balayé par les intérêts individuels des utilisateurs.

† Si les clients sont des navigateurs avec des caches de navigateur, la figure 7-9 montre techniquement une hiérarchie de cache à trois niveaux.

‡ Les caches parents doivent être plus grands pour contenir les documents populaires auprès d'un plus grand nombre d'utilisateurs et plus performants, car ils reçoivent le trafic global de nombreux enfants, dont les intérêts peuvent être divers.

Topologies de cache

Figure 7-8. Les caches publics partagés peuvent réduire le trafic réseau.

caches, mais chaque proxy intermédiaire impose une pénalité de performance qui peut devenir perceptible à mesure que la chaîne de proxy devient longue.*

Maillages de cache, routage de contenu et peering

Certains architectes réseau construisent des maillages de cache complexes plutôt que de simples hiérarchies de cache. Les caches proxy des maillages de cache communiquent entre eux de manière plus sophistiquée et prennent des décisions de communication dynamiques, en choisissant les caches parents avec lesquels communiquer ou en choisissant de contourner complètement les caches et de se diriger vers le serveur d'origine. Ces caches proxy peuvent être décrits comme des routeurs de contenu, car ils prennent des décisions de routage concernant l'accès, la gestion et la diffusion du contenu.

Les caches conçus pour le routage de contenu au sein de maillages de cache peuvent effectuer toutes les opérations suivantes (entre autres) :

• Choisissez entre un cache parent ou un serveur d'origine de manière dynamique, en fonction de l'URL.

• Sélectionnez un cache parent particulier de manière dynamique, en fonction de l'URL.

* En pratique, les architectes réseau tentent de se limiter à deux ou trois proxys consécutifs. Cependant, une nouvelle génération de serveurs proxy hautes performances pourrait réduire le problème de la longueur de la chaîne de proxy.

Figure 7-9. Accès aux documents dans une hiérarchie de cache à deux niveaux

• Recherchez dans les caches de la zone locale une copie en cache avant d'accéder à un cache parent.

• Autoriser d'autres caches à accéder à des parties de leur contenu mis en cache, mais ne pas autoriser

Internet transite par leur cache.

Ces relations plus complexes entre les caches permettent à différentes organisations de s'appairer entre elles, connectant leurs caches pour un bénéfice mutuel. Les caches prenant en charge l'appairage sélectif sont appelés caches frères (Figure 7-10). HTTP ne prenant pas en charge les caches frères, il a été étendu à l'aide de protocoles tels que le protocole ICP (Internet Cache Protocol) et le protocole HyperText Caching Protocol.

(HTCP). Nous aborderons ces protocoles au chapitre 20.

Étapes de traitement du cache

Les caches proxy commerciaux modernes sont assez complexes. Ils sont conçus pour être très performants et prendre en charge les fonctionnalités avancées de HTTP et d'autres technologies. Cependant, malgré quelques subtilités, le fonctionnement de base d'un cache web est globalement simple. Une séquence de traitement de cache basique pour un message HTTP GET se compose de sept étapes.

(illustré dans la figure 7-11) :

1. Réception : le cache lit le message de demande arrivant sur le réseau.

2. Analyse : le cache analyse le message, en extrayant l'URL et les en-têtes.

Étapes de traitement du cache

Figure 7-10. Caches frères

3. Recherche : le cache vérifie si une copie locale est disponible et, dans le cas contraire, récupère une copie

(et le stocke localement).

4. Vérification de fraîcheur : le cache vérifie si la copie mise en cache est suffisamment récente et, si ce n'est pas le cas, demande au serveur des mises à jour.

5. Création de réponse : le cache crée un message de réponse avec les nouveaux en-têtes et le corps mis en cache.

6. Envoi : le cache renvoie la réponse au client via le réseau.

7. Journalisation : le cache crée éventuellement une entrée de fichier journal décrivant la transaction.

Étape 1 : Réception

À l'étape 1, le cache détecte l'activité sur une connexion réseau et lit les données entrantes. Les caches hautes performances lisent simultanément les données de plusieurs connexions entrantes et commencent à traiter la transaction avant l'arrivée du message complet.

Étape 2 : Analyse

Ensuite, le cache analyse le message de requête en plusieurs parties et place les en-têtes dans des structures de données faciles à manipuler. Cela permet au logiciel de mise en cache de traiter et de manipuler plus facilement les champs d'en-tête.*

* L'analyseur est également chargé de normaliser les parties de l'en-tête afin que les différences mineures, comme l'utilisation des majuscules ou les formats de date alternatifs, soient toutes traitées de manière équivalente. De plus, comme certains messages de requête contiennent une URL absolue complète et d'autres une URL relative et un en-tête d'hôte, l'analyseur masque généralement ces détails (voir « URL relatives » au chapitre 2).

Figure 7-11. Traitement d'un nouveau résultat de cache

Étape 3 : Recherche

À l'étape 3, le cache récupère l'URL et recherche une copie locale. Cette copie peut être stockée en mémoire, sur un disque local, voire sur un autre ordinateur à proximité. Les caches professionnels utilisent des algorithmes rapides pour déterminer si un objet est disponible dans le cache local. Si le document n'est pas disponible localement, il peut être récupéré depuis le serveur d'origine ou un proxy parent, ou renvoyer un message d'erreur, selon la situation et la configuration.

L'objet mis en cache contient le corps de la réponse du serveur et les en-têtes de la réponse d'origine, ce qui permet de renvoyer les en-têtes serveur corrects lors d'une occurrence dans le cache. Il inclut également des métadonnées permettant de comptabiliser sa durée de vie.

resté dans la cache, combien de fois il a été utilisé, etc.*

Étape 4 : Vérification de la fraîcheur

HTTP permet aux caches de conserver des copies des documents du serveur pendant un certain temps. Pendant ce temps, le document est considéré comme « à jour » et le cache peut le servir sans contacter le serveur. En revanche, si la copie en cache est restée trop longtemps, au-delà de la limite de fraîcheur du document, l'objet est considéré comme « périmé » et le cache doit être réinitialisé.

* Les caches sophistiqués conservent également une copie des en-têtes de réponse client d'origine qui ont généré la réponse du serveur, à utiliser dans la négociation de contenu HTTP/1.1 (voir chapitre 17).

Étapes de traitement du cache

Revalidez le document auprès du serveur pour vérifier toute modification avant de le diffuser. Les en-têtes de requête envoyés par un client au cache compliquent encore les choses, car ils peuvent forcer le cache à revalider ou à éviter toute validation.

HTTP utilise un ensemble de règles très complexes pour vérifier la fraîcheur, rendues plus complexes par le grand nombre d'options de

configuration prises en charge par les produits de cache et par la nécessité d'interagir avec des normes de fraîcheur non HTTP. Nous consacrerons la majeure partie du reste de ce chapitre à l'explication des calculs de fraîcheur.

Étape 5 : Création de la réponse

Comme nous souhaitons que la réponse mise en cache semble provenir du serveur d'origine, le cache utilise les en-têtes de réponse du serveur mis en cache comme point de départ. Ces en-têtes de base sont ensuite modifiés et complétés par le cache.

Le cache est chargé d'adapter les en-têtes aux besoins du client. Par exemple, le serveur peut renvoyer une réponse HTTP/1.0 (voire HTTP/0.9), tandis que le client attend une réponse HTTP/1.1 ; dans ce cas, le cache doit traduire les en-têtes en conséquence. Les caches intègrent également des informations sur la fraîcheur du cache (en-têtes Cache-Control, Age et Expires) et incluent souvent un en-tête Via pour signaler qu'un cache proxy a traité la requête.

Notez que le cache ne doit pas ajuster l'en-tête Date. Cet en-tête représente la date de génération initiale de l'objet sur le serveur d'origine.

Étape 6 : Envoi

Une fois les en-têtes de réponse prêts, le cache renvoie la réponse au client. Comme tous les serveurs proxy, un cache proxy doit gérer la connexion avec le client. Les caches hautes performances s'efforcent d'envoyer les données efficacement, évitant souvent la copie du contenu du document entre le stockage local et les tampons d'E/S réseau.

Étape 7 : Journalisation

La plupart des caches conservent des fichiers journaux et des statistiques sur leur utilisation. Après chaque transaction, le cache met à jour les statistiques en comptant le nombre de requêtes réussies et manquées (et d'autres indicateurs pertinents) et insère une entrée dans un fichier journal indiquant le type de requête, l'URL et le déroulement de l'opération.

Les formats de journaux de cache les plus courants sont Squid et Netscape, mais de nombreux produits de cache permettent de créer des fichiers journaux personnalisés. Nous détaillons les formats de fichiers journaux au chapitre 21.

Organigramme du traitement du cache

La figure 7-12 montre, sous une forme simplifiée, comment un cache traite une demande d'OBTENTION d'une URL.*

Figure 7-12. Organigramme de la requête GET du cache

Garder les copies à jour

Les copies en cache peuvent ne pas être cohérentes avec les documents du serveur. Après tout, les documents évoluent au fil du temps. Les rapports peuvent changer tous les mois, les journaux en ligne quotidiennement, les données financières peuvent changer toutes les quelques secondes. Les caches seraient inutiles s'ils servaient toujours d'anciennes données. Les données en cache doivent conserver une certaine cohérence avec celles du serveur.

HTTP inclut des mécanismes simples permettant de maintenir la cohérence des données mises en cache avec les serveurs, sans que ces derniers aient à mémoriser les caches contenant des copies de leurs documents. HTTP appelle ces mécanismes simples « expiration des documents » et « revalidation du serveur ».

Expiration du document

HTTP permet à un serveur d'origine d'associer une date d'expiration à chaque document, grâce aux en-têtes HTTP Cache-Control et Expires (Figure 7-13). À l'instar de la date d'expiration d'un litre de lait, ces en-têtes déterminent la durée pendant laquelle un contenu doit être considéré comme frais.

* La revalidation et la récupération d'une ressource comme indiqué dans la Figure 7-12 peuvent être effectuées en une seule étape avec une requête conditionnelle (voir « Revalidation avec des méthodes conditionnelles »).

HTTP/1.0 200 OK

Date : sam. 29 juin 2002, 14 h 30 GMT

Type de contenu : texte/brut

Longueur du contenu : 67

Expire : ven. 5 juil. 2002, 05:00:00 GMT

Soldes du Jour de l'Indépendance chez Joe's Hardware Venez magasiner avec nous aujourd'hui ! HTTP/1.0 200 OK

Date : sam. 29 juin 2002, 14 h 30 GMT

Type de contenu : texte/brut

Longueur du contenu : 67

Contrôle du cache : max-age=484 200

Vente du jour de l'indépendance chez Joe's Hardware Venez magasiner avec nous aujourd'hui !

(a) En-tête Expires (b) Cache-Control : en-tête max-age

Figure 7-13. En-têtes Expires et Cache Control

Jusqu'à l'expiration d'un document en cache, le cache peut en servir la copie aussi souvent qu'il le souhaite, sans jamais contacter le serveur, sauf si, bien sûr, une requête client contient des en-têtes empêchant la fourniture d'une ressource mise en cache ou non validée. Cependant, une fois le document en cache expiré, le cache doit vérifier auprès du serveur si le document a été modifié et, le cas échéant, obtenir une nouvelle copie (avec une nouvelle date d'expiration).

Dates d'expiration et âges

Les serveurs spécifient les dates d'expiration à l'aide des en-têtes de réponse HTTP/1.0+ Expires ou HTTP/1.1 Cache-Control: max-age, qui accompagnent le corps de la réponse. Les en-têtes Expires et Cache-Control: max-age ont une fonction similaire, mais le nouvel en-tête Cache-Control est préféré, car il utilise une heure relative plutôt qu'une date absolue. Les dates absolues dépendent du bon réglage des horloges des ordinateurs.

Le tableau 7-2 répertorie les en-têtes de réponse d’expiration.

Tableau 7-2. En-têtes de réponse d'expiration

Description de l'en-tête

Cache-Control : max-age La valeur max-age définit l'âge maximal du document : le temps légal maximal écoulé (en secondes) entre le moment où un document est généré pour la première fois et celui où il ne peut plus être considéré comme suffisamment récent pour être diffusé.

Contrôle du cache : max-age=484 200

Expire : spécifie une date d'expiration absolue. Si la date d'expiration est passée, le document n'est plus à jour.

Expire : ven. 5 juil. 2002, 05:00:00 GMT

Imaginons que nous sommes le 29 juin 2002 à 9h30, heure normale de l'Est (HNE), et que la quincaillerie de Joe se prépare pour les soldes du 4 juillet (dans seulement cinq jours). Joe souhaite créer une page web spéciale sur son serveur et la faire expirer à minuit HNE le soir du 5 juillet 2002. Si le serveur de Joe utilise les anciens en-têtes Expires, le message de réponse du serveur (Figure 7-13a) pourrait inclure cet en-tête :*

Expire : ven. 5 juil. 2002, 05:00:00 GMT

* Veuillez noter que toutes les dates et heures HTTP sont exprimées en heure moyenne de Greenwich (GMT). L'heure GMT correspond au méridien de Greenwich (longitude 0°) qui passe par Greenwich, au Royaume-Uni. L'heure GMT a cinq heures d'avance sur l'heure normale de l'Est des États-Unis ; minuit EST correspond donc à 5 h 00 GMT.

Si le serveur de Joe utilise les en-têtes Cache-Control: max-age plus récents, le message de réponse du serveur (Figure 7-13b) peut contenir cet en-tête :

Contrôle du cache : max-age=484 200

Au cas où cela ne serait pas immédiatement évident, 484 200 correspond au nombre de secondes entre la date du jour, le 29 juin 2002 à 9h30 HNE, et la date de fin de la vente, le 5 juillet 2002 à minuit. Il reste 134,5 heures (environ 5 jours) avant la fin de la vente. Avec 3 600 secondes par heure, il reste 484 200 secondes avant la fin de la vente.

Revalidation du serveur

L'expiration d'un document en cache ne signifie pas qu'il est différent de celui du serveur d'origine ; cela signifie simplement qu'il est temps de le vérifier. C'est ce qu'on appelle la « revalidation du serveur », ce qui signifie que le cache doit demander au serveur d'origine si le document a été modifié :

• Si la revalidation montre que le contenu a changé, le cache obtient une nouvelle copie du document, la stocke à la place des anciennes données et envoie le document au client.

• Si la revalidation montre que le contenu n'a pas changé, le cache obtient uniquement de nouveaux en-têtes, y compris une nouvelle date d'expiration, et met à jour les en-têtes dans le cache.

C'est un système efficace. Le cache n'a pas besoin de vérifier la fraîcheur d'un document à chaque requête ; il doit le revalider auprès du serveur uniquement après expiration du document. Cela permet d'économiser le trafic serveur et d'améliorer le temps de réponse des utilisateurs, sans diffuser de contenu obsolète.

Le protocole HTTP nécessite un cache fonctionnant correctement pour renvoyer l'un des éléments suivants :

• Une copie en cache qui est « suffisamment récente »

• Une copie en cache qui a été revalidée auprès du serveur pour garantir qu'elle est toujours à jour

• Un message d'erreur si le serveur d'origine avec lequel revalider est en panne*

• Une copie en cache, avec un avertissement joint indiquant qu'elle pourrait être incorrecte

Revalidation avec des méthodes conditionnelles

Les méthodes conditionnelles de HTTP optimisent la revalidation. HTTP permet à un cache d'envoyer une requête GET conditionnelle au serveur d'origine, lui demandant de renvoyer le corps de l'objet

uniquement si le document est différent de la copie actuellement en cache. De cette manière, la vérification de la fraîcheur et la récupération de l'objet sont combinées en une seule requête GET conditionnelle. Les requêtes GET conditionnelles sont initiées par l'ajout d'en-têtes conditionnels spéciaux aux messages de requête GET. Le serveur web ne renvoie l'objet que si la condition est vraie.

* Si le serveur d'origine n'est pas accessible, mais que le cache doit être revalidé, celui-ci doit renvoyer une erreur ou un avertissement décrivant l'échec de communication. Dans le cas contraire, les pages d'un serveur supprimé peuvent rester dans les caches réseau pendant une durée indéterminée.

HTTP définit cinq en-têtes de requête conditionnelle. Les deux plus utiles pour la revalidation du cache sont If-Modified-Since et If-None-Match.* Tous les en-têtes conditionnels commencent par le préfixe « If- ». Le tableau 7-3 répertorie les en-têtes de réponse conditionnelle utilisés pour la revalidation du cache.

Tableau 7-3. Deux en-têtes conditionnels utilisés pour la revalidation du cache

Description de l'en-tête

Si-Modifié-Depuis :

<date> Exécute la méthode demandée si le document a été modifié depuis la date spécifiée. Cette méthode est utilisée conjointement avec l'en-tête de réponse du serveur Last-Modified pour récupérer le contenu uniquement si celui-ci a été modifié depuis la version en cache.

If-None-Match : <tags> Au lieu de rechercher la date de dernière modification, le serveur peut ajouter des balises spéciales (voir « ETag » dans l'annexe C) au document, qui agissent comme des numéros de série. L'en-tête If-None-Match exécute la méthode demandée si les balises mises en cache diffèrent de celles du document du serveur.

Si-Modifié-Depuis : Date de revalidation

L'en-tête de revalidation de cache le plus courant est If-Modified-Since. Les requêtes de revalidation If-Modified-Since sont souvent appelées requêtes « IMS ». Les requêtes IMS indiquent au serveur de n'exécuter la requête que si la ressource a changé depuis une certaine date :

• Si le document a été modifié depuis la date spécifiée, la condition If-Modified-Since est vraie et l'exécution GET réussit normalement. Le nouveau document est renvoyé au cache, accompagné de nouveaux en-têtes contenant, entre autres informations, une nouvelle date d'expiration.

• Si le document n'a pas été modifié depuis la date spécifiée, la condition est fausse et un court message de réponse 304 « Non

modifié » est renvoyé au client, sans corps de document, pour plus d'efficacité.† Les en-têtes sont renvoyés dans la réponse ; toutefois, seuls ceux qui nécessitent une mise à jour par rapport à l'original doivent être renvoyés. Par exemple, l'en-tête « Content-Type » n'a généralement pas besoin d'être envoyé, car il n'a généralement pas changé. Une nouvelle date d'expiration est généralement envoyée.

L'en-tête If-Modified-Since fonctionne conjointement avec l'en-tête de réponse du serveur Last-Modified. Le serveur d'origine associe la date de dernière modification aux documents servis. Lorsqu'un cache souhaite revalider un document mis en cache, il inclut un en-tête If-Modified-Since avec la date de dernière modification de la copie mise en cache :

If-Modified-Since : <date de dernière modification mise en cache>

* D'autres en-têtes conditionnels incluent If-Unmodified-Since (utile pour les transferts de documents partiels, lorsque vous devez vous assurer que le document est inchangé avant d'en récupérer un autre morceau), If-Range (pour prendre en charge la mise en cache des documents incomplets) et If-Match (utile pour le contrôle de la concurrence lors de la gestion des serveurs Web).

† Si un ancien serveur ne reconnaissant pas l'en-tête If-Modified-Since reçoit la requête conditionnelle, il l'interprète comme un GET normal. Dans ce cas, le système fonctionnera toujours, mais son efficacité sera réduite en raison de la transmission inutile de données de document inchangées.

Si le contenu a changé entre-temps, la date de dernière modification sera différente et le serveur d'origine renverra le nouveau document. Sinon, le serveur constatera que la date de dernière modification du cache correspond à la date de dernière modification actuelle du document serveur et renverra une réponse 304 « Non modifié ».

Par exemple, comme illustré à la figure 7-14, si votre cache revalide l'annonce des soldes du 4 juillet de Joe's Hardware le 3 juillet, vous recevrez une réponse « Non modifié » (figure 7-14a). En revanche, si votre cache revalide le document après la fin des soldes, le 5 juillet à minuit, il recevra un nouveau document, car le contenu du serveur a changé (figure 7-14b).

Figure 7-14. Les revalidations If-Modified-Since renvoient 304 si inchangé ou 200 avec un nouveau corps si modifié.

Notez que certains serveurs web n'implémentent pas If-Modified-Since comme véritable comparaison de date. Ils effectuent plutôt une comparaison de chaîne entre la date IMS et la date de dernière modification. Ainsi, la sémantique se comporte comme « si la dernière modification n'est pas à cette date précise » au lieu de « si la

modification a eu lieu depuis cette date ». Cette sémantique alternative fonctionne bien pour l'expiration du cache, lorsque la date de dernière modification est utilisée comme un numéro de série, mais elle empêche les clients d'utiliser l'en-tête If-Modified-Since à des fins temporelles.

If-None-Match : revalidation des balises d'entité

Il existe certaines situations dans lesquelles la revalidation de la date de dernière modification n'est pas adéquate :

• Certains documents peuvent être réécrits périodiquement (par exemple, à partir d'un processus d'arrière-plan), mais contiennent souvent les mêmes données. Les dates de modification changeront, même si le contenu est resté le même.

• Certains documents peuvent avoir changé, mais seulement d'une manière qui n'est pas suffisamment importante pour justifier que des caches dans le monde entier rechargent les données (par exemple, des modifications d'orthographe ou de commentaires).

• Certains serveurs ne peuvent pas déterminer avec précision les dates de dernière modification de leurs pages.

• Pour les serveurs qui diffusent des documents qui changent à des intervalles inférieurs à une seconde (par exemple, les moniteurs en temps réel), la granularité d'une seconde des dates de modification peut ne pas être adéquate.

Pour contourner ces problèmes, HTTP permet de comparer les « identifiants de version » des documents, appelés balises d'entité (ETags). Les balises d'entité sont des étiquettes arbitraires (chaînes entre guillemets) attachées au document. Elles peuvent contenir un numéro de série ou un nom de version du document, ou encore une somme de contrôle ou une autre empreinte du contenu du document.

Lorsque l'éditeur modifie un document, il peut modifier la balise d'entité du document pour représenter cette nouvelle version. Les caches peuvent ensuite utiliser l'en-tête conditionnel If-None-Match pour obtenir une nouvelle copie du document si les balises d'entité ont changé.

Dans la figure 7-15, le cache contient un document avec la balise d'entité « v2.6 ». Il revalide auprès du serveur d'origine et demande un nouvel objet uniquement si la balise « v2.6 » ne correspond plus. Dans la figure 7-15, la balise correspond toujours, ce qui renvoie une réponse 304 « Non modifié ».

Figure 7-15. La fonction If-None-Match est revalidée, car la balise d'entité correspond toujours.

Si la balise d'entité sur le serveur avait changé (peut-être vers « v3.0 »), le serveur renverrait le nouveau contenu dans une réponse 200 OK, ainsi que le contenu et la nouvelle ETag.

Plusieurs balises d'entité peuvent être incluses dans un en-tête If-None-Match, pour indiquer au serveur que le cache possède déjà des copies d'objets avec ces balises d'entité :

Si-Aucune-Correspondance : « v2.6 »

Si-Aucune-Correspondance : « v2.4 », « v2.5 », « v2.6 »

If-None-Match : « foobar », « A34FAC0095 », « Profils in Courage »

Validateurs faibles et forts

Les caches utilisent des balises d'entité pour déterminer si la version mise en cache est à jour par rapport au serveur (de la même manière qu'ils utilisent les dates de dernière modification). Ainsi, les balises d'entité et les dates de dernière modification sont toutes deux des validateurs de cache.

Les serveurs peuvent parfois souhaiter autoriser des modifications superficielles ou insignifiantes sur des documents sans invalider toutes les copies en cache. HTTP/1.1 prend en charge les « validateurs faibles », qui permettent au serveur de revendiquer une équivalence « suffisamment bonne » même si le contenu a légèrement changé.

Les validateurs forts changent à chaque modification du contenu. Les validateurs faibles autorisent certaines modifications de contenu, mais changent généralement lorsque le sens du contenu change. Certaines opérations ne peuvent pas être effectuées avec des validateurs faibles (comme les récupérations conditionnelles de plages partielles). Les serveurs identifient donc les validateurs faibles par le préfixe « W/ ».

ETag : W/« v2.6 »

Si-Aucune-Correspondance : W/« v2.6 »

Une balise d'entité forte doit être modifiée dès que la valeur de l'entité associée change, quelle qu'en soit la nature. Une balise d'entité faible doit être modifiée dès que l'entité associée change de manière significative sur le plan sémantique.

Notez qu'un serveur d'origine doit éviter de réutiliser une valeur de balise d'entité forte spécifique pour deux entités différentes, ou de réutiliser une valeur de balise d'entité faible spécifique pour deux entités sémantiquement différentes. Les entrées du cache peuvent persister pendant des périodes arbitrairement longues, indépendamment des dates d'expiration ; il serait donc inapproprié de s'attendre à ce qu'un cache ne tente plus jamais de valider une entrée à l'aide d'un validateur obtenu précédemment.

Quand utiliser les balises d'entité et les dates de dernière modification

Les clients HTTP/1.1 doivent utiliser un validateur de balises d'entité si un serveur renvoie une balise d'entité. Si le serveur renvoie uniquement une valeur de dernière modification, le client peut utiliser la validation If-Modified-Since. Si une balise d'entité et une date de dernière modification sont disponibles, le client doit utiliser les deux schémas de revalidation, permettant ainsi aux caches HTTP/1.0 et HTTP/1.1 de répondre correctement.

Les serveurs d'origine HTTP/1.1 doivent envoyer un validateur de balise d'entité, sauf s'il est impossible d'en générer un. Il peut s'agir d'une balise d'entité faible plutôt que forte, si les validateurs faibles présentent des avantages. Il est également préférable d'envoyer une valeur de dernière modification.

Si un cache ou un serveur HTTP/1.1 reçoit une requête avec des en-têtes conditionnels If-Modified-Since et des en-têtes conditionnels de balise d'entité, il ne doit pas renvoyer une réponse 304 Not Modified, sauf si cela est cohérent avec tous les champs d'en-tête conditionnels de la requête.

Contrôle de la mise en cache

HTTP définit plusieurs méthodes permettant à un serveur de spécifier la durée de mise en cache d'un document avant son expiration. Par ordre de priorité décroissante, le serveur peut :

• Attachez un en-tête Cache-Control: no-store à la réponse.

• Attachez un en-tête Cache-Control: no-cache à la réponse.

• Attachez un en-tête Cache-Control: must-revalidate à la réponse.

• Attachez un en-tête Cache-Control: max-age à la réponse.

• Joignez un en-tête de date d’expiration à la réponse.

• Ne joignez aucune information d'expiration, laissant le cache déterminer sa propre date d'expiration heuristique.

Cette section décrit les en-têtes de contrôle du cache. La section suivante, « Configuration des contrôles du cache », explique comment attribuer différentes informations de cache à différents contenus.

En-têtes de réponse sans cache et sans stockage

HTTP/1.1 propose plusieurs moyens de limiter la mise en cache des objets, ou leur diffusion, afin de préserver leur actualité. Les en-têtes no-store et no-cache empêchent les caches de diffuser des objets non vérifiés :

Cache-Control : no-store

Contrôle du cache : pas de cache

Pragma : sans cache

Une réponse marquée « no-store » empêche le cache d'en faire une copie. Un cache transmet généralement une réponse no-store au client, puis supprime l'objet, comme le ferait un serveur proxy sans mise en cache.

Une réponse marquée « no-cache » peut être stockée dans le cache local. Elle ne peut simplement pas être servie au client depuis le cache sans revalidation préalable de sa fraîcheur auprès du serveur d'origine. Un meilleur nom pour cet en-tête serait « do-notserve-from-cache-without-revalidation ».

L'en-tête Pragma: no-cache est inclus dans HTTP/1.1 pour assurer la rétrocompatibilité avec HTTP/1.0 et les versions ultérieures. Les applications HTTP 1.1 doivent utiliser Cache-Control: no-cache, sauf pour les applications HTTP 1.0, qui ne comprennent que Pragma: no-cache.*

En-têtes de réponse Max-Age

L'en-tête Cache-Control: max-age indique le nombre de secondes écoulées depuis sa sortie du serveur pour qu'un document soit considéré comme récent. Il existe également un en-tête smaxage (notez l'absence de tiret dans « maxage ») qui fonctionne comme max-age, mais ne s'applique qu'aux caches partagés (publics) :

Contrôle du cache : max-age=3600

Contrôle du cache : s-maxage=3600

Les serveurs peuvent demander que les caches ne mettent pas en cache un document ou qu'ils s'actualisent à chaque accès en définissant le vieillissement maximal sur zéro :

Contrôle du cache : max-age=0

Contrôle du cache : s-maxage=0

En-têtes de réponse expirés

L'en-tête obsolète Expires spécifie une date d'expiration réelle plutôt qu'une durée en secondes. Les concepteurs HTTP ont ensuite décidé que, compte tenu des horloges désynchronisées ou incorrectes de nombreux serveurs, il serait préférable de représenter l'expiration en secondes écoulées plutôt qu'en temps absolu. Une durée de vie comparable peut être calculée en calculant la différence en secondes entre la valeur d'expiration et la valeur de date :

Expire : ven. 5 juil. 2002, 05:00:00 GMT

Certains serveurs renvoient également un en-tête de réponse « Expires : 0 » pour tenter de systématiquement faire expirer les documents, mais cette syntaxe est illégale et peut entraîner des

problèmes avec certains logiciels. Vous devriez essayer de prendre en charge cette construction en entrée, mais ne la générez pas.

En-têtes de réponse à revalider

Les caches peuvent être configurés pour traiter les objets périmés afin d'améliorer les performances. Si un serveur d'origine souhaite que les caches respectent scrupuleusement les informations d'expiration, il peut y joindre un contrôle de cache :

Cache-Control : doit-revalider

L'en-tête de réponse Cache-Control: must-revalidate indique aux caches qu'ils ne peuvent pas servir une copie obsolète de cet objet sans une revalidation préalable auprès du serveur d'origine. Les caches peuvent toujours servir de nouvelles copies. Si le serveur d'origine est indisponible lorsqu'un cache tente une vérification de fraîcheur de type must-revalidate, le cache doit renvoyer une erreur 504 Gateway Timeout.

* Pragma no-cache n'est techniquement valable que pour les requêtes HTTP, mais il est largement utilisé comme en-tête d'extension pour les requêtes et les réponses HTTP.

Contrôle de la mise en cache

Expiration heuristique

Si la réponse ne contient ni en-tête Cache-Control: max-age ni en-tête Expires, le cache peut calculer un âge maximal heuristique. N'importe quel algorithme peut être utilisé, mais si l'âge maximal obtenu est supérieur à 24 heures, un en-tête d'avertissement d'expiration heuristique (Avertissement 13) doit être ajouté aux en-têtes de réponse. À notre connaissance, peu de navigateurs mettent cet avertissement à la disposition des utilisateurs.

Un algorithme heuristique d'expiration populaire, l'algorithme LM-Factor, peut être utilisé si le document contient une date de dernière modification. L'algorithme LM-Factor utilise cette date pour estimer la volatilité d'un document. Voici la logique :

• Si un document mis en cache a été modifié pour la dernière fois dans un passé lointain, il peut s'agir d'un document stable et moins susceptible de changer soudainement. Il est donc plus sûr de le conserver plus longtemps dans le cache.

• Si le document mis en cache a été modifié récemment, il change probablement fréquemment, nous devrions donc le mettre en cache seulement un court instant avant de le revalider avec le serveur.

L'algorithme LM-Factor calcule le temps écoulé entre le moment où le cache communique avec le serveur et celui où ce dernier déclare la dernière modification du document. Il prend une fraction de ce temps

intermédiaire et l'utilise comme durée de conservation dans le cache. Voici un exemple de pseudo-code Perl pour l'algorithme LM-Factor :

$time_since_modify = max(0, $server_Date - $server_Last_Modified);

$server_freshness_limit = int($time_since_modify * $lm_factor);

La figure 7-16 illustre graphiquement la période de fraîcheur du facteur LM. La ligne hachurée indique la période de fraîcheur, avec un facteur LM de 0,2.

Figure 7-16. Calcul d'une période de fraîcheur à l'aide de l'algorithme LM-Factor

En général, les utilisateurs limitent les périodes de fraîcheur heuristiques afin d'éviter qu'elles ne deviennent excessivement longues. Une semaine est généralement la durée habituelle, bien que les sites plus conservateurs utilisent un jour.

Enfin, si vous ne disposez pas non plus de date de dernière modification, le cache ne dispose pas de beaucoup d'informations. Les caches attribuent généralement une durée de conservation par défaut (une heure ou un jour en général) aux documents sans indication de durée de conservation. Les caches plus conservateurs choisissent parfois une durée de conservation de 0 pour ces documents heuristiques, obligeant le cache à vérifier que les données sont toujours à jour avant chaque diffusion à un client.

Un dernier point concernant les calculs heuristiques de fraîcheur : ils sont plus courants qu'on ne le pense. De nombreux serveurs d'origine ne génèrent toujours pas d'en-têtes Expires et Max-Age. Choisissez soigneusement les valeurs d'expiration par défaut de votre cache !

Contraintes de fraîcheur du client

Les navigateurs web disposent d'un bouton « Actualiser » ou « Recharger » pour forcer l'actualisation du contenu, qui peut être obsolète dans le cache du navigateur ou du proxy. Ce bouton émet une requête GET avec des en-têtes de requête Cache-control supplémentaires qui forcent une revalidation ou une récupération inconditionnelle depuis le serveur. Le comportement précis de l'actualisation dépend du navigateur, du document et des configurations de cache.

Les clients utilisent les en-têtes de requête Cache-Control pour renforcer ou assouplir les contraintes d'expiration. Ils peuvent également rendre l'expiration plus stricte pour les applications nécessitant des documents très récents (comme le bouton d'actualisation manuel). D'autre part, les clients peuvent également assouplir les exigences de fraîcheur afin d'améliorer les performances,

la fiabilité ou les coûts. Le tableau 7-4 résume les directives de requête Cache-Control.

Tableau 7-4. Directives de requête Cache-Control

Objectif de la directive

Cache-Control : max-stale

Cache-Control : max-stale =

<s> Le cache est libre de servir un document obsolète. Si le paramètre <s> est spécifié, le document ne doit pas être obsolète plus longtemps que ce délai. Cela assouplit les règles de mise en cache.

Cache-Control : min-fresh =

<s> Le document doit être encore à jour pendant au moins <s> secondes à l'avenir. Cela rend les règles de mise en cache plus strictes.

Cache-Control : max-age = <s> Le cache ne peut pas renvoyer un document mis en cache depuis plus de <s> secondes. Cette directive renforce les règles de mise en cache, sauf si la directive max-stale est également définie, auquel cas l'âge peut dépasser son délai d'expiration.

Contrôle du cache : pas de cache

Pragma : no-cache Ce client n'acceptera pas une ressource mise en cache à moins qu'elle n'ait été revalidée.

Cache-Control : no-store Le cache doit supprimer toute trace du document du stockage dès que possible, car il peut contenir des informations sensibles.

Cache-Control : only-if-cached Le client souhaite une copie uniquement si elle est dans le cache.

Précautions

L'expiration des documents n'est pas un système parfait. Si un éditeur attribue accidentellement une date d'expiration trop éloignée, les modifications qu'il doit apporter au document n'apparaîtront pas forcément dans tous les caches avant l'expiration du document.* Pour cette raison,

* L'expiration d'un document est une technique de « durée de vie » utilisée dans de nombreux protocoles Internet, comme DNS. DNS, comme HTTP, rencontre des difficultés si vous publiez une date d'expiration lointaine et que vous constatez ensuite qu'une modification est nécessaire. Cependant, HTTP offre des mécanismes permettant à un client de contourner et de forcer un rechargement, contrairement à DNS.

Contrôle de la mise en cache

De nombreux éditeurs n'utilisent pas de dates d'expiration lointaines. De plus, de nombreux éditeurs n'utilisent même pas de dates d'expiration, ce qui rend difficile pour les caches de savoir combien de temps un document restera à jour.

Configuration des contrôles du cache

Les serveurs web proposent différents mécanismes pour définir les en-têtes HTTP de contrôle du cache et d'expiration. Dans cette section, nous aborderons brièvement la prise en charge du contrôle du cache par le célèbre serveur web Apache. Consultez la documentation de votre serveur web pour plus de détails.

Contrôle des en-têtes HTTP avec Apache

Le serveur web Apache propose plusieurs mécanismes permettant de définir les en-têtes de contrôle du cache HTTP. Nombre de ces mécanismes ne sont pas activés par défaut ; vous devez les activer (parfois en obtenant au préalable des modules d'extension Apache). Voici une brève description de certaines fonctionnalités d'Apache :

en-têtes de mod

Le module mod_headers vous permet de définir des en-têtes individuels. Une fois ce module chargé, vous pouvez enrichir les fichiers de configuration Apache avec des directives pour définir des en-têtes HTTP individuels. Vous pouvez également utiliser ces paramètres en combinaison avec les expressions régulières et les filtres d'Apache pour associer des en-têtes à du contenu individuel. Voici un exemple de configuration permettant de marquer tous les fichiers HTML d'un répertoire comme non cachables :

<Fichiers *.html>

Ensemble d'en-têtes Cache-control no-cache </Files> mod_expires

Le module mod_expires fournit une logique de programmation pour générer automatiquement des en-têtes Expires avec les dates d'expiration correctes. Ce module vous permet de définir des dates d'expiration après le dernier accès ou la dernière modification d'un document. Il vous permet également d'attribuer différentes dates d'expiration à différents types de fichiers et d'utiliser des descriptions détaillées, comme « accès plus 1 mois », pour décrire la mise en cache. Voici quelques exemples :

Expire par défaut A3600

Expire par défaut M86400

ExpiresDefault « accès plus 1 semaine » ExpiresByType text/html « modification plus 2 jours 6 heures 12 minutes »

mod_cern_meta

Le module mod_cern_meta permet d'associer un fichier d'en-têtes HTTP à des objets spécifiques. En activant ce module, vous créez un ensemble de « métafichiers », un pour chaque document à contrôler, et ajoutez les en-têtes souhaités à chaque métafichier.

Contrôle de la mise en cache HTML via HTTP-EQUIV

Les en-têtes de réponse du serveur HTTP servent à transmettre les informations d'expiration des documents et de contrôle du cache. Les serveurs Web interagissent avec les fichiers de configuration pour attribuer les en-têtes de contrôle du cache appropriés aux documents servis.

Pour permettre aux auteurs d'attribuer plus facilement des informations d'en-tête HTTP aux documents HTML diffusés sans interagir avec les fichiers de configuration du serveur web, HTML 2.0 a défini la balise <META HTTP-EQUIV>. Cette balise facultative se trouve en haut d'un document HTML et définit les en-têtes HTTP à associer au document. Voici un exemple de balise <META HTTP-EQUIV> définie pour signaler qu'un document HTML n'est pas cachable :

```
<HTML>
 <TÊTE>
  <TITLE>Mon document</TITLE>
  <META HTTP-EQUIV="Cache-control" CONTENT="no-cache">
 </HEAD> ...
```

Cette balise HTTP-EQUIV était initialement destinée aux serveurs web. Ces derniers étaient censés analyser le code HTML à la recherche des balises <META HTTP-EQUIV> et insérer les en-têtes requis dans la réponse HTTP, comme indiqué dans la RFC HTML 1866 :

Un serveur HTTP peut utiliser ces informations pour traiter le document. Il peut notamment inclure un champ d'en-tête dans les réponses aux requêtes pour ce document : le nom de l'en-tête est extrait de la valeur de l'attribut HTTP-EQUIV et la valeur de l'en-tête est extraite de la valeur de l'attribut CONTENT.

Malheureusement, peu de serveurs Web et de proxys prennent en charge cette fonctionnalité facultative en raison de la charge supplémentaire du serveur, des valeurs statiques et du fait qu'il ne prend en charge que HTML et non les nombreux autres types de fichiers.

Cependant, certains navigateurs analysent et respectent les balises HTTP-EQUIV dans le contenu HTML, traitant les en-têtes intégrés comme de véritables en-têtes HTTP (Figure 7-17). Ceci est regrettable, car les navigateurs HTML prenant en charge HTTP-EQUIV peuvent

appliquer des règles de contrôle de cache différentes de celles des caches proxy intermédiaires. Cela entraîne un comportement d'expiration du cache déroutant.

En général, les balises <META HTTP-EQUIV> ne permettent pas de contrôler la mise en cache des documents. Le seul moyen sûr de communiquer les requêtes de contrôle de cache pour les documents est d'utiliser les en-têtes HTTP envoyés par un serveur correctement configuré.

Algorithmes détaillés

La spécification HTTP fournit un algorithme détaillé, mais légèrement obscur et souvent déroutant, pour calculer le vieillissement des documents et la fraîcheur du cache. Dans cette section, nous détaillerons les algorithmes de calcul de la fraîcheur HTTP (le losange « Assez frais ? » de la figure 7-12) et expliquerons leur motivation.

Figure 7-17. Les balises HTTP-EQUIV posent problème, car la plupart des logiciels les ignorent.

Cette section sera particulièrement utile aux lecteurs travaillant sur les mécanismes internes du cache. Pour illustrer la formulation de la spécification HTTP, nous utiliserons du pseudo-code Perl. Si les détails complexes des formules d'expiration du cache ne vous intéressent pas, n'hésitez pas à ignorer cette section.

Âge et fraîcheur Durée de vie

Pour déterminer si un document en cache est suffisamment récent pour être diffusé, le cache doit calculer seulement deux valeurs : l'âge et la durée de vie de la copie en cache. Si l'âge d'une copie en cache est inférieur à sa durée de vie, la copie est suffisamment récente pour être diffusée. En Perl :

$is_fresh_enough = ($age < $freshness_lifetime);

L'âge d'un document correspond au temps total écoulé depuis son envoi depuis le serveur (ou sa dernière revalidation par le serveur).* Étant donné qu'un cache peut ne pas savoir si une réponse provient d'un cache en amont ou d'un serveur, il ne peut pas présumer que le document est neuf. Il doit déterminer l'âge du document, soit à partir d'un en-tête Age explicite (de préférence), soit en traitant l'en-tête Date généré par le serveur.

La durée de vie d'un document indique l'âge maximal d'une copie en cache avant qu'elle ne soit plus suffisamment récente pour être diffusée aux clients. Elle prend en compte la date d'expiration du document et les éventuelles modifications de durée de vie demandées par le client.

Certains clients peuvent accepter des documents légèrement obsolètes (en utilisant l'en-tête Cache-Control : max-stale). D'autres clients peuvent refuser des documents qui deviendront obsolètes prochainement (en utilisant l'en-tête Cache-Control : min-fresh). Le cache combine les informations d'expiration du serveur avec les exigences de fraîcheur du client pour déterminer la durée de vie maximale.

Calcul de l'âge

L'âge de la réponse correspond au temps total écoulé depuis son émission par le serveur (ou sa revalidation). Cet âge inclut le temps passé par la réponse dans les routeurs et les passerelles Internet, le temps de stockage dans les caches intermédiaires et le temps passé dans votre cache.

L'exemple 7-1 fournit un pseudo-code pour le calcul de l'âge.

Exemple 7-1. L'algorithme de calcul de l'âge HTTP/1.1 calcule l'âge global d'un document mis en cache

```
$apparent_age = max(0, $time_got_response - $Date_header_value);

$corrected_apparent_age = max($apparent_age, $Age_header_value);

$response_delay_estimate = ($time_got_response -
$time_issued_request);

$age_when_document_arrived_at_our_cache =

   $corrected_apparent_age + $response_delay_estimate;

$how_long_copy_has_been_in_our_cache = $current_time -
$time_got_response;

$age = $age_quand_le_document_est_arrivé_à_notre_cache +

     $combien_de_temps_la_copie_est_elle_restée_dans_notre_cache;
```

Le calcul de l'âge HTTP est un peu complexe, mais le concept de base est simple. Les caches peuvent déterminer l'âge de la réponse à son arrivée en examinant les en-têtes Date ou Age. Ils peuvent également noter la durée de conservation du document dans le cache local. L'ensemble de ces valeurs représente l'âge total de la réponse. HTTP utilise des astuces pour tenter de compenser les décalages d'horloge et les retards réseau, mais le calcul de base est assez simple :

```
$age = $age_quand_le_document_est_arrivé_à_notre_cache +

     $combien_de_temps_la_copie_est_elle_restée_dans_notre_cache;
```

* N'oubliez pas que le serveur dispose toujours de la version la plus récente de tout document.

Un cache permet de déterminer assez facilement la durée de conservation locale d'une copie en cache (une simple question de

comptabilité), mais il est plus difficile de déterminer l'âge d'une réponse lorsqu'elle arrive au cache, car tous les serveurs ne disposent pas d'horloges synchronisées et nous ignorons où se trouve la réponse. L'algorithme complet de calcul de l'âge tente de remédier à ce problème.

L'âge apparent est basé sur l'en-tête Date

Si tous les ordinateurs partageaient la même horloge exacte, l'âge d'un document en cache correspondrait simplement à son « âge apparent » : l'heure actuelle moins l'heure d'envoi du document par le serveur. L'heure d'envoi du serveur correspond simplement à la valeur de l'en-tête Date. Le calcul initial le plus simple de l'âge utiliserait simplement l'âge apparent :

$apparent_age = $time_got_response - $Date_header_value;

$age_when_document_arrived_at_our_cache = $apparent_age;

Malheureusement, toutes les horloges ne sont pas parfaitement synchronisées. Les horloges client et serveur peuvent différer de plusieurs minutes, voire de plusieurs heures ou jours si elles sont mal réglées.*

Les applications web, en particulier les proxys de mise en cache, doivent être préparées à interagir avec des serveurs dont les valeurs d'horloge diffèrent considérablement. Ce problème est appelé décalage d'horloge : la différence entre les réglages d'horloge de deux ordinateurs. À cause de ce décalage, l'âge apparent est parfois inexact, voire négatif.

Si l'âge est négatif, nous le mettons simplement à zéro. Nous pourrions également vérifier que l'âge apparent n'est pas excessivement élevé, mais des âges apparents élevés pourraient être corrects. Nous pourrions communiquer avec un cache parent qui a mis le document en cache pendant longtemps (le cache stocke également l'en-tête Date d'origine) :

$apparent_age = max(0, $time_got_response - $Date_header_value);

$age_when_document_arrived_at_our_cache = $apparent_age;

Veuillez noter que l'en-tête Date décrit la date d'origine du serveur. Les proxys et les caches ne doivent pas modifier cette date !

Calculs d'âge houblon par houblon

Nous pouvons donc éliminer les âges négatifs causés par le décalage d'horloge, mais nous ne pouvons pas faire grand-chose contre la perte globale de précision due à ce décalage. HTTP/1.1 tente de contourner l'absence d'horloges universellement synchronisées en demandant à chaque appareil d'accumuler le vieillissement relatif dans un en-tête Age, lorsqu'un document transite par des proxys et des caches.

De cette façon, aucune comparaison d'horloge de bout en bout entre serveurs n'est nécessaire.

La valeur de l'en-tête Age augmente à mesure que le document transite par les proxys. Les applications compatibles HTTP/1.1 doivent augmenter la valeur de l'en-tête Age au moment où le document est transmis.

* La spécification HTTP recommande que les clients, les serveurs et les proxys utilisent un protocole de synchronisation horaire tel que NTP pour garantir une base de temps cohérente.

installé dans chaque application et en transit réseau. Chaque application intermédiaire peut facilement calculer l'heure de résidence du document grâce à son horloge locale.

Cependant, tout périphérique non HTTP/1.1 dans la chaîne de réponse ne reconnaîtra pas l'en-tête Age et le transmettra sans modification ou le supprimera. Ainsi, tant que HTTP/1.1 ne sera pas universellement adopté, l'en-tête Age sous-estimera l'âge relatif.

Les valeurs d'âge relatives sont utilisées en complément du calcul de l'âge basé sur la date. La plus prudente des deux estimations est retenue, car la valeur de date interserveur ou la valeur calculée par l'âge peut être sous-estimée (la plus prudente étant l'âge le plus ancien). De cette façon, HTTP tolère également les erreurs dans les en-têtes d'âge, tout en privilégiant le contenu le plus récent :

$apparent_age = max(0, $time_got_response - $Date_header_value);

$corrected_apparent_age = max($apparent_age, $Age_header_value);

$age_when_document_arrived_at_our_cache = $corrected_apparent_age;

Compenser les retards du réseau

Les transactions peuvent être lentes. C'est la principale raison de la mise en cache. Cependant, pour les réseaux très lents ou les serveurs surchargés, le calcul de l'âge relatif peut sous-estimer considérablement l'âge des documents si ceux-ci restent longtemps bloqués dans les embouteillages du réseau ou du serveur.

L'en-tête Date indique quand le document a quitté le serveur d'origine*, mais n'indique pas le temps de transit jusqu'au cache. Si le document est passé par une longue chaîne de proxys et de caches parents, le délai réseau peut être important†.

Il n'existe pas de méthode simple pour mesurer le délai réseau unidirectionnel du serveur au cache, mais il est plus facile de mesurer le délai aller-retour. Un cache sait quand le document a été demandé et quand il est arrivé. HTTP/1.1 corrige de manière prudente ces délais réseau en additionnant la totalité du délai aller-retour. Ce délai cache-

serveur-cache est une surestimation du délai serveur-cache, mais il est prudent. S'il est erroné, il ne fera que rendre les documents plus anciens qu'ils ne le sont en réalité et entraînera des revalidations inutiles. Voici comment le calcul est effectué :

$apparent_age = max(0, $time_got_response - $Date_header_value);

$corrected_apparent_age = max($apparent_age, $Age_header_value);

$response_delay_estimate = ($time_got_response - $time_issued_request);

$age_when_document_arrived_at_our_cache =

$corrected_apparent_age + $response_delay_estimate;

* Notez que si le document provient d'un cache parent et non d'un serveur d'origine, l'en-tête Date reflétera la date du serveur d'origine et non celle du cache parent.

† En pratique, cela ne devrait pas dépasser quelques dizaines de secondes (sinon les utilisateurs abandonneront), mais les concepteurs HTTP voulaient essayer de prendre en charge l'expiration précise même des objets à courte durée de vie.

Algorithme complet de calcul de l'âge

La section précédente a montré comment calculer l'âge d'un document HTTP lorsqu'il arrive en cache. Une fois cette réponse stockée dans le cache, son âge augmente. Lorsqu'une requête arrive pour le document en cache, nous devons connaître la durée de sa présence afin de pouvoir calculer son âge actuel :

$age = $age_quand_le_document_est_arrivé_à_notre_cache +

$combien_de_temps_la_copie_est_elle_restée_dans_notre_cache;

Et voilà ! Nous obtenons ainsi l'algorithme complet de calcul d'âge HTTP/1.1 présenté dans l'exemple 7-1. Il s'agit d'une simple comptabilité : nous savons quand le document est arrivé dans le cache ($time_got_response) et quand la requête actuelle est arrivée (maintenant), donc l'heure de résidence ne représente que la différence. Tout cela est illustré graphiquement dans la figure 7-18.

Figure 7-18. L'âge d'un document mis en cache inclut le temps de résidence sur le réseau et dans le cache.

Calcul de la durée de vie de la fraîcheur

Rappelons que nous cherchons à déterminer si un document mis en cache est suffisamment récent pour être diffusé à un client. Pour répondre à cette question, nous devons déterminer l'âge du document mis en cache et calculer sa durée de vie en fonction des contraintes du

serveur et du client. Nous venons d'expliquer comment calculer l'âge ; passons maintenant aux durées de vie.

La durée de vie d'un document indique l'âge qu'il peut atteindre avant de ne plus être suffisamment récent pour être diffusé à un client particulier. Cette durée dépend des contraintes du serveur et du client. Le serveur peut disposer d'informations sur le taux de modification des publications du document. Les rapports classés, très stables, peuvent rester à jour pendant des années. Les périodiques peuvent n'être à jour que jusqu'à la prochaine publication prévue, soit la semaine prochaine ou demain matin à 6h.

Les clients peuvent avoir d'autres exigences. Ils peuvent accepter du contenu légèrement obsolète, s'il est plus rapide, ou avoir besoin du contenu le plus récent possible. Les caches servent les utilisateurs. Nous devons respecter leurs demandes.

Algorithme complet de fraîcheur du serveur

L'exemple 7-2 présente un algorithme Perl permettant de calculer les limites de fraîcheur du serveur. Il renvoie l'âge maximal qu'un document peut atteindre sans être interrompu par le serveur.

Exemple 7-2. Calcul de la contrainte de fraîcheur du serveur

limite_de_fraîcheur_du_sous-serveur

```
{ local($heuristic,$server_freshness_limit,$time_since_last_modify);
$heuristic = 0;

si ($Max_Age_value_set)

{

$server_freshness_limit = $Max_Age_value;

}

elsif ($Expires_value_set)

{

$server_freshness_limit = $Expires_value - $Date_value;

}

elsif ($Last_Modified_value_set)

{

$time_since_last_modify = max(0, $Date_value -
$Last_Modified_value);

$server_freshness_limit =
int($temps_depuis_la_dernière_modification * $lm_factor);

$heuristique = 1;
```

```
} autre

{ $server_freshness_limit = $default_cache_min_lifetime;

$heuristique = 1;

}

si ($heuristique)

{ si ($server_freshness_limit > $default_cache_max_lifetime) {
$server_freshness_limit = $default_cache_max_lifetime; } si
($server_freshness_limit < $default_cache_min_lifetime) {
$server_freshness_limit = $default_cache_min_lifetime; }

}

retour($server_freshness_limit);

}
```

Voyons maintenant comment le client peut outrepasser la limite d'âge du document spécifiée par le serveur. L'exemple 7-3 présente un algorithme Perl qui prend une limite d'âge du serveur et la modifie en fonction des contraintes du client. Il renvoie l'âge maximal qu'un document peut atteindre tout en étant servi par le cache sans revalidation.

Exemple 7-3. Calcul de la contrainte de fraîcheur du client

```
sous-client_modifié_limite_de_fraîcheur

{

   $age_limit = server_freshness_limit(); ## Extrait de l'exemple 7-2

   si ($Max_Stale_value_set)

   {

      si ($Max_Stale_value == $INT_MAX)

      { $age_limit = $INT_MAX; } sinon

      { $age_limit = server_freshness_limit() + $Max_Stale_value; }

   }

   si ($Min_Fresh_value_set)

   {

      $age_limit = min($age_limit, server_freshness_limit() -
$Min_Fresh_value_set);

   }

   si ($Max_Age_value_set)
```

```
  {
    $age_limit = min($age_limit, $Max_Age_value);
  }
}
```

L'ensemble du processus implique deux variables : l'âge du document et sa limite de fraîcheur. Le document est considéré comme suffisamment récent si son âge est inférieur à la limite de fraîcheur. L'algorithme de l'exemple 7-3 prend simplement la limite de fraîcheur du serveur et la fait varier en fonction des contraintes supplémentaires du client. Nous espérons que cette section a clarifié les subtils algorithmes d'expiration décrits dans les spécifications HTTP.

Caches et publicité

Si vous êtes arrivé jusqu'ici, vous avez compris que les caches améliorent les performances et réduisent le trafic. Vous savez que les caches peuvent aider les utilisateurs et leur offrir une meilleure expérience, et vous savez qu'ils peuvent aider les opérateurs réseau à réduire leur trafic.

Le dilemme de l'annonceur

On pourrait également s'attendre à ce que les fournisseurs de contenu apprécient les caches. Après tout, si les caches étaient omniprésents, ils n'auraient pas besoin d'acheter de gros serveurs web multiprocesseurs pour répondre à la demande, ni de payer des frais de service réseau exorbitants pour fournir sans cesse les mêmes données à leurs utilisateurs. Mieux encore,

Les caches permettent aux articles et publicités accrocheurs d'apparaître plus rapidement et plus efficacement sur l'écran des internautes, les incitant à consommer davantage de contenu et à voir davantage de publicités. Et c'est exactement ce que recherchent les fournisseurs de contenu : plus de visibilité et plus de publicités !

Mais c'est là que le bât blesse. De nombreux fournisseurs de contenu sont rémunérés grâce à la publicité, notamment à chaque fois qu'une publicité est présentée à un utilisateur (peut-être une fraction de centime ou deux, mais cela représente une somme considérable si vous diffusez un million de publicités par jour !). Et c'est là tout le problème des caches : ils peuvent masquer le nombre réel d'accès au serveur d'origine. Si la mise en cache était parfaite, un serveur d'origine pourrait ne recevoir aucun accès HTTP, car ils seraient absorbés par les caches Internet. Mais si vous êtes rémunéré au nombre d'accès, vous ne vous réjouirez pas.

La réponse de l'éditeur

Aujourd'hui, les annonceurs utilisent toutes sortes de techniques de « cache-busting » pour s'assurer que les caches ne volent pas leur flux de visites. Ils ajoutent des en-têtes sans cache à leur contenu, diffusent des publicités via des passerelles CGI et réécrivent les URL des publicités sur chaque page.

accéder.

Ces techniques de contournement du cache ne concernent pas uniquement les caches proxy. En fait, elles ciblent aujourd'hui principalement le cache activé dans chaque navigateur web. Malheureusement, en s'efforçant de maintenir leur flux de visites, certains fournisseurs de contenu réduisent les effets positifs de la mise en cache sur leur site.

Dans un monde idéal, les fournisseurs de contenu laisseraient les caches absorber leur trafic, et ceux-ci leur indiqueraient le nombre de visites reçues. Aujourd'hui, les caches peuvent y parvenir de plusieurs manières.

Une solution consiste à configurer les caches pour qu'ils soient revalidés auprès du serveur d'origine à chaque accès. Cela envoie un hit au serveur d'origine à chaque accès, mais ne transfère généralement aucune donnée de corps. Bien sûr, cela ralentit la transaction.*

Migration des journaux

Une solution idéale ne nécessiterait pas l'envoi de hits au serveur. Après tout, le cache peut conserver un journal de tous les hits. Les caches pourraient simplement distribuer les journaux de hits aux serveurs. En fait, certains grands fournisseurs de cache traitent et livrent manuellement les journaux de cache aux fournisseurs de contenu influents pour satisfaire ces derniers.

* Certains caches prennent en charge une variante de cette revalidation : ils effectuent une requête GET ou HEAD conditionnelle en arrière-plan. L'utilisateur ne perçoit pas le délai, mais la requête déclenche un accès hors ligne au serveur d'origine. Il s'agit d'une amélioration, mais elle alourdit la charge des caches et augmente considérablement le trafic sur le réseau.

Caches et publicité

Malheureusement, les journaux d'accès sont volumineux, ce qui les rend difficiles à déplacer. De plus, les journaux de cache ne sont ni standardisés ni organisés de manière à séparer les journaux envoyés vers les différents fournisseurs de contenu. De plus, ils posent des problèmes d'authentification et de confidentialité.

Des propositions ont été faites pour des schémas de redistribution des logs efficaces (et moins efficaces). Aucun n'est suffisamment développé

pour être adopté par les éditeurs de logiciels web. Nombre d'entre eux sont extrêmement complexes et nécessitent des partenariats commerciaux pour réussir.* Plusieurs entreprises ont été lancées pour développer une infrastructure de soutien à la récupération des revenus publicitaires.

Mesure des hits et limitation de l'utilisation

La RFC 2227, « Simple Hit-Metering and Usage-Limiting for HTTP », définit un schéma beaucoup plus simple. Ce protocole ajoute un nouvel en-tête à HTTP, appelé Meter, qui transmet périodiquement aux serveurs le nombre d'accès à des URL spécifiques. Ainsi, les serveurs reçoivent des mises à jour régulières des caches concernant le nombre d'accès aux documents mis en cache.

De plus, le serveur peut contrôler le nombre de fois que les documents peuvent être servis depuis le cache, ou un délai d'expiration, avant que le cache ne doive envoyer un rapport au serveur. C'est ce qu'on appelle la limitation d'utilisation ; elle permet aux serveurs de contrôler la quantité d'utilisation d'une ressource mise en cache avant qu'elle ne doive envoyer un rapport au serveur d'origine.

Nous décrirons la RFC 2227 en détail dans le chapitre 21.

Pour plus d'informations

Pour plus d'informations sur la mise en cache, reportez-vous à :

http://www.w3.org/Protocols/rfc2616/rfc2616.txt

RFC 2616, « Protocole de transfert hypertexte », par R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Mastinter, P. Leach et T. Berners-Lee.

Mise en cache Web

Duane Wessels, O'Reilly & Associates, Inc.

http://search.ietf.org/rfc/rfc3040.txt

RFC 3040, « Taxonomie de la réplication et de la mise en cache du Web Internet ».

Serveurs proxy Web

Ari Luotonen, Livres informatiques de Prentice Hall.

http://search.ietf.org/rfc/rfc3143.txt

RFC 3143, « Problèmes connus de proxy/mise en cache HTTP ».
http://www.squid-cache.org

Cache du proxy Web Squid.

* Plusieurs entreprises ont tenté de développer des solutions globales pour la mise en cache et la journalisation intégrées.

Chapitre 8Ceci est le titre du livre CHAPITRE 8

Points d'intégration : passerelles, tunnels et relais

Le Web s'est révélé être un formidable outil de diffusion de contenu. Au fil du temps, les utilisateurs sont passés de la simple mise en ligne de documents statiques à la volonté de partager des ressources toujours plus complexes, comme du contenu de bases de données ou des pages HTML générées dynamiquement. Les applications HTTP, comme les navigateurs web, ont offert aux utilisateurs un moyen unifié d'accéder au contenu sur Internet.

HTTP est également devenu un élément fondamental pour les développeurs d'applications, qui s'appuient sur d'autres protocoles (par exemple, pour acheminer ou relayer le trafic d'autres protocoles à travers les pare-feu d'entreprise, en l'encapsulant dans HTTP). HTTP est utilisé comme protocole pour toutes les ressources du Web, et c'est aussi un protocole utilisé par d'autres applications et protocoles applicatifs pour accomplir leurs tâches.

fait.

Ce chapitre examine de manière générale certaines des méthodes mises au point par les développeurs pour utiliser HTTP afin d'accéder à différentes ressources et examine comment les développeurs utilisent HTTP comme cadre pour activer d'autres protocoles et la communication des applications.

Dans ce chapitre, nous discutons :

• Passerelles, qui interfacent HTTP avec d'autres protocoles et applications

• Interfaces d'application, qui permettent à différents types d'applications Web de communiquer entre elles

• Tunnels, qui vous permettent d'envoyer du trafic non HTTP via des connexions HTTP

• Relais, qui sont un type de proxy HTTP simplifié utilisé pour transférer des données un saut à la fois

Passerelles

L'histoire des extensions et interfaces HTTP a été dictée par les besoins des utilisateurs. Lorsque le désir de mettre des ressources plus complexes sur le Web est apparu, il est rapidement devenu évident qu'aucune application ne pouvait gérer à elle seule toutes les ressources imaginables.

Pour contourner ce problème, les développeurs ont imaginé une passerelle pouvant servir d'interpréteur, permettant d'accéder à la ressource de manière abstraite. Une passerelle est le lien entre les ressources et les applications. Une application peut demander (via HTTP ou une autre interface définie) à une passerelle de traiter la requête, et celle-ci peut fournir une réponse. La passerelle peut communiquer le langage de requête à la base de données ou générer du contenu dynamique, agissant comme un portail : une requête est reçue et une réponse est émise.

La figure 8-1 illustre une sorte de passerelle de ressources. Ici, le serveur Joe's Hardware fait office de passerelle vers le contenu de la base de données. Notez que le client demande simplement une ressource via HTTP, et le serveur Joe's Hardware s'interface avec une passerelle pour y accéder.

Figure 8-1. Magie de la passerelle

Certaines passerelles traduisent automatiquement le trafic HTTP vers d’autres protocoles, de sorte que les clients HTTP peuvent interagir avec d’autres applications sans que les clients aient besoin de connaître d’autres protocoles (Figure 8-2).

La figure 8-2 montre trois exemples de passerelles :

• Dans la figure 8-2a, la passerelle reçoit des requêtes HTTP pour les URL FTP. Elle ouvre ensuite les connexions FTP et envoie les commandes appropriées au serveur FTP. Le document est renvoyé via HTTP, accompagné de l'URL HTTP correcte.

en-têtes.

• Dans la figure 8-2b, la passerelle reçoit une requête web chiffrée via SSL, la déchiffre* et transmet une requête HTTP normale au serveur de destination. Ces accélérateurs de sécurité peuvent être placés directement devant les serveurs web (généralement dans les mêmes locaux) pour fournir un chiffrement haute performance aux serveurs d'origine.

* La passerelle doit disposer des certificats de serveur appropriés installés.

Figure 8-2. Trois exemples de passerelles Web

• Dans la figure 8-2c, la passerelle connecte les clients HTTP aux programmes d'application côté serveur, via une API de passerelle de serveur d'applications. Lorsque vous effectuez des achats en ligne, consultez la météo ou consultez le cours de la bourse, vous accédez à des passerelles de serveur d'applications.

Passerelles côté client et côté serveur

Les passerelles Web utilisent HTTP d'un côté et un protocole différent de l'autre côté.* Les passerelles sont décrites par leurs protocoles côté client et côté serveur, séparés par une barre oblique :

<protocole-client>/<protocole-serveur>

Ainsi, une passerelle reliant des clients HTTP à des serveurs de news NNTP est une passerelle HTTP/NNTP. Nous utilisons les termes « passerelle côté serveur » et « passerelle côté client » pour décrire le côté de la passerelle concerné par la conversion :

• Les passerelles côté serveur parlent HTTP avec les clients et un protocole étranger avec les serveurs (HTTP/*).

• Les passerelles côté client parlent des protocoles étrangers avec les clients et HTTP avec les serveurs (*/HTTP).

* Les proxys web qui convertissent entre différentes versions de HTTP sont comparables à des passerelles, car ils exécutent une logique sophistiquée pour négocier entre les parties. Mais comme ils utilisent HTTP des deux côtés, ils sont techniquement des proxys.

Passerelles

Passerelles de protocole

Vous pouvez diriger le trafic HTTP vers des passerelles de la même manière que vous dirigez le trafic vers des proxys. Le plus souvent, vous configurez explicitement les navigateurs pour utiliser des passerelles, intercepter le trafic de manière transparente ou configurer les passerelles comme des proxys inverses.

La figure 8-3 présente les boîtes de dialogue permettant de configurer un navigateur pour l'utilisation de passerelles FTP côté serveur. Dans la configuration illustrée, le navigateur est configuré pour utiliser gw1.joeshardware.com comme passerelle HTTP/FTP pour toutes les URL FTP. Au lieu d'envoyer des commandes FTP à un serveur FTP, le navigateur enverra des commandes HTTP à la passerelle HTTP/FTP gw1.joes-hardware.com sur le port 8080.

Figure 8-3. Configuration d'une passerelle HTTP/FTP

Le résultat de cette configuration de passerelle est illustré à la figure 8-4. Le trafic HTTP normal n'est pas affecté ; il continue de circuler directement vers les serveurs d'origine. Cependant, les requêtes d'URL FTP sont envoyées à la passerelle gw1.joes-hardware.com via des requêtes HTTP. La passerelle effectue les transactions FTP pour le compte du client et lui renvoie les résultats via HTTP.

Les sections suivantes décrivent les types courants de passerelles : convertisseurs de protocole de serveur, passerelles de sécurité côté serveur, passerelles de sécurité côté client et serveurs d'applications.

HTTP/* : passerelles Web côté serveur

Les passerelles Web côté serveur convertissent les requêtes HTTP côté client en un protocole étranger, lorsque les requêtes transitent vers le serveur d'origine (voir Figure 8-5).

Dans la figure 8-5, la passerelle reçoit une requête HTTP pour une ressource FTP :

ftp://ftp.irs.gov/pub/00-index.txt

Figure 8-5. La passerelle HTTP/FTP convertit les requêtes HTTP en requêtes FTP.

La passerelle ouvre une connexion FTP sur le port FTP du serveur d'origine (port 21) et utilise le protocole FTP pour récupérer l'objet. La passerelle effectue les opérations suivantes :

• Envoie les commandes USER et PASS pour se connecter au serveur

• Émet la commande CWD pour passer au répertoire approprié sur le serveur

• Définit le type de téléchargement sur ASCII

• Récupère l'heure de la dernière modification du document avec MDTM

• Indique au serveur d'attendre une demande de récupération de données passive à l'aide de PASV

• Demande la récupération de l'objet à l'aide de RETR

• Ouvre une connexion de données au serveur FTP sur un port renvoyé sur le canal de contrôle ; dès que le canal de données est ouvert, le contenu de l'objet revient vers la passerelle

Passerelles de protocole

Une fois la récupération terminée, l'objet sera envoyé au client dans une réponse HTTP.

HTTP/HTTPS : passerelles de sécurité côté serveur

Les passerelles peuvent être utilisées pour renforcer la confidentialité et la sécurité d'une organisation, en chiffrant toutes les requêtes web entrantes. Les clients peuvent naviguer sur le Web en utilisant leurs identifiants habituels.

HTTP, mais la passerelle cryptera automatiquement les sessions de l'utilisateur (Figure 8-6).

Figure 8-6. Passerelle de sécurité HTTP/HTTPS entrante

HTTPS/HTTP : passerelles d'accélération de la sécurité côté client

Récemment, les passerelles HTTPS/HTTP sont devenues populaires comme accélérateurs de sécurité. Ces passerelles HTTPS/HTTP se trouvent en amont du serveur web, généralement comme passerelle d'interception invisible ou proxy inverse. Elles reçoivent le trafic HTTPS sécurisé, le déchiffrent et envoient des requêtes HTTP classiques au serveur web (Figure 8-7).

Figure 8-7. Passerelle d'accélération de sécurité HTTPS/HTTP

Ces passerelles intègrent souvent un matériel de déchiffrement spécifique permettant de déchiffrer le trafic sécurisé bien plus efficacement que le serveur d'origine, allégeant ainsi la charge de ce dernier. Comme ces passerelles transmettent du trafic non chiffré entre elles et le serveur d'origine, il est important de veiller à la sécurité du réseau entre elles.

Passerelles de ressources

Jusqu'à présent, nous avons parlé des passerelles qui connectent les clients et les serveurs sur un réseau. Cependant, la forme la plus courante de passerelle, le serveur d'applications, combine le serveur de destination et la passerelle en un seul serveur. Les serveurs d'applications sont des passerelles côté serveur qui communiquent en HTTP avec le client et se connectent à un programme d'application côté serveur (voir Figure 8-8).

Figure 8-8. Un serveur d'applications connecte des clients HTTP à des applications back-end arbitraires.

Dans la figure 8-8, deux clients se connectent à un serveur d'applications via HTTP. Cependant, au lieu de renvoyer des fichiers depuis le serveur, celui-ci transmet les requêtes aux applications exécutées sur le serveur via une interface de programmation d'applications (API) de passerelle :

• La requête du client A est reçue et, en fonction de l'URI, envoyée via une API à une application d'appareil photo numérique. L'image résultante est intégrée dans un message de réponse HTTP et renvoyée au client pour affichage dans son navigateur.

• L'URI du client B est celle d'une application de commerce électronique. Les requêtes du client B sont envoyées via l'API de la passerelle serveur au logiciel de commerce électronique, et les résultats sont renvoyés au navigateur. Le logiciel de commerce électronique interagit avec le client, guidant l'utilisateur à travers une séquence de pages HTML pour finaliser son achat.

La première API populaire pour les passerelles applicatives était l'interface CGI (Common Gateway Interface). CGI est un ensemble standardisé d'interfaces que les serveurs web utilisent pour lancer des programmes en réponse à des requêtes HTTP pour des URL spécifiques, collecter le résultat du programme et le renvoyer dans des réponses HTTP. Ces dernières années, les serveurs web commerciaux ont fourni des interfaces plus sophistiquées pour connecter les serveurs web aux applications.

Passerelles de ressources

Les premiers serveurs Web étaient des créations assez simples, et l’approche simple adoptée pour mettre en œuvre une interface pour les passerelles est restée inchangée jusqu’à aujourd’hui.

Lorsqu'une requête arrive pour une ressource nécessitant une passerelle, le serveur génère l'application d'assistance pour la traiter. Les données nécessaires lui sont transmises. Il s'agit souvent de la requête complète ou de la requête que l'utilisateur souhaite exécuter sur la base de données (à partir de la chaîne de requête de l'URL ; voir chapitre 2).

Il renvoie ensuite une réponse ou des données de réponse au serveur, qui les transmet au client. Le serveur et la passerelle étant des applications distinctes, les responsabilités sont clairement définies. La figure 8-9 illustre les mécanismes de base des interactions entre le serveur et la passerelle.

Figure 8-9. Mécanique de l'application de passerelle serveur

Ce protocole simple (requête entrante, transfert et réponse) est l'essence même de l'une des interfaces d'extension de serveur les plus anciennes et les plus courantes, CGI.

Interface de passerelle commune (CGI)

L'interface Common Gateway a été la première extension serveur, et reste probablement la plus utilisée. Elle est utilisée sur tout le Web pour des tâches telles que le HTML dynamique, le traitement des cartes de crédit et l'interrogation de bases de données.

Les applications CGI étant indépendantes du serveur, elles peuvent être implémentées dans presque tous les langages, y compris Perl, Tcl, C et divers langages shell. De plus, la simplicité de CGI permet à presque

tous les serveurs HTTP de le prendre en charge. Les mécanismes de base du modèle CGI sont illustrés à la figure 8-9.

Le traitement CGI est invisible pour les utilisateurs. Du point de vue du client, il s'agit simplement d'une requête normale. Il ignore totalement la procédure de transfert entre le serveur et l'application CGI. Le seul indice qu'une application CGI pourrait être impliquée serait la présence des lettres « cgi » et peut-être « ? » dans l'URL.

Alors, le CGI est formidable, non ? Eh bien, oui et non. Il fournit une forme simple et fonctionnelle de lien entre les serveurs et pratiquement tous les types de ressources, gérant toutes les traductions nécessaires. L'interface est également élégante : elle protège le serveur des extensions boguées (si l'extension était intégrée au serveur lui-même, elle pourrait provoquer une erreur pouvant entraîner le plantage du serveur).

Cependant, cette séparation a un impact négatif sur les performances. La charge de travail nécessaire pour générer un nouveau processus à chaque requête CGI est importante, ce qui limite les performances des serveurs utilisant CGI et sollicite les ressources de la machine. Pour contourner ce problème, une nouvelle forme de CGI, judicieusement baptisée Fast CGI, a été développée. Cette interface imite CGI, mais fonctionne comme un démon persistant, éliminant ainsi la baisse de performances liée à la configuration et à la suppression d'un nouveau processus à chaque requête.

API d'extension de serveur

Le protocole CGI offre un moyen simple d'interfacer des interpréteurs externes avec des serveurs HTTP standard. Mais que faire si vous souhaitez modifier le comportement du serveur lui-même ou simplement optimiser ses performances ? Pour répondre à ces deux besoins, les développeurs de serveurs ont mis au point des API d'extension serveur, qui offrent une interface puissante permettant aux développeurs web d'interfacer directement leurs propres modules avec un serveur HTTP. Ces API d'extension permettent aux programmeurs d'intégrer leur propre code au serveur ou de remplacer complètement un composant du serveur par le leur.

La plupart des serveurs populaires proposent une ou plusieurs API d'extension aux développeurs. Ces extensions étant souvent liées à l'architecture du serveur, la plupart sont spécifiques à un type de serveur. Microsoft, Netscape, Apache et d'autres serveurs disposent tous d'interfaces API permettant aux développeurs de modifier le comportement du serveur ou de personnaliser les interfaces avec différentes ressources. Ces interfaces personnalisées offrent une interface puissante aux développeurs.

L'extension serveur FrontPage (FPSE) de Microsoft est un exemple d'extension serveur. Elle prend en charge les services de publication

web pour les auteurs FrontPage. FPSE est capable d'interpréter les commandes d'appel de procédure distante (RPC) envoyées par les clients FrontPage. Ces commandes sont superposées à HTTP (plus précisément, à la méthode HTTP POST). Pour plus de détails, voir « Extensions serveur FrontPage pour la prise en charge de la publication » au chapitre 19.

Interfaces d'application et services Web

Nous avons évoqué les passerelles de ressources comme moyen de communication entre les serveurs web et les applications. Plus généralement, avec la multiplication des services proposés par les applications web, il apparaît clairement que HTTP peut constituer une base pour relier les applications entre elles. L'un des points les plus délicats du câblage applicatif est la négociation de l'interface de protocole entre les deux applications afin qu'elles puissent échanger des données ; cette opération se fait souvent application par application.

Interfaces d'application et services Web

Pour fonctionner ensemble, les applications doivent généralement échanger des informations plus complexes que celles exprimées dans les en-têtes HTTP. Le chapitre 19 présente quelques exemples d'extension de HTTP ou de superposition de protocoles HTTP pour échanger des informations personnalisées. « Extensions serveur FrontPage pour la prise en charge de la publication » du chapitre 19 traite de la superposition de RPC sur des messages HTTP POST, et « WebDAV et création collaborative » traite de l'ajout de XML aux en-têtes HTTP.

La communauté Internet a développé un ensemble de normes et de protocoles permettant aux applications web de communiquer entre elles. Ces normes sont communément appelées « services web », bien que ce terme puisse également désigner des applications web autonomes (blocs de construction). Le principe des services web n'est pas nouveau, mais ils constituent un nouveau mécanisme de partage d'informations entre applications. Les services web s'appuient sur des technologies web standard, telles que HTTP.

Les services Web échangent des informations via XML via SOAP. Le langage XML (Extensible Markup Language) permet de créer et d'interpréter des informations personnalisées sur un objet de données. Le protocole SOAP (Simple Object Access Protocol) est une norme permettant d'ajouter des informations XML aux messages HTTP*.

Tunnels

Nous avons abordé les différentes manières d'utiliser HTTP pour accéder à divers types de ressources (via des passerelles) et pour permettre la communication entre applications. Dans cette section,

nous examinerons une autre utilisation de HTTP : les tunnels web, qui permettent d'accéder à des applications utilisant des protocoles non HTTP via des applications HTTP.

Les tunnels web permettent d'acheminer du trafic non HTTP via des connexions HTTP, permettant ainsi à d'autres protocoles de se greffer sur HTTP. L'utilisation la plus courante des tunnels web est d'intégrer du trafic non HTTP à une connexion HTTP, afin de pouvoir le faire passer à travers des pare-feu autorisant uniquement le trafic web.

Établissement de tunnels HTTP avec CONNECT

Les tunnels Web sont établis à l'aide de la méthode CONNECT de HTTP. Le protocole CONNECT ne fait pas partie de la spécification HTTP/1.1 principale†, mais il s'agit d'une extension largement implémentée. Les spécifications techniques sont disponibles dans le projet de spécification Internet d'Ari Luotonen, désormais expiré, « Tunneling TCP based protocols through Web proxy servers », ou dans son ouvrage « Web Proxy Servers », tous deux cités à la fin de ce chapitre.

* Pour plus d'informations, voir http://www.w3.org/TR/2001/WD-soap12-part0-20011217/. L'ouvrage « Programmation de services Web avec SOAP », de Doug Tidwell, James Snell et Pavel Kulchenko (O'Reilly), est également une excellente source d'informations sur le protocole SOAP.

† La spécification HTTP/1.1 réserve la méthode CONNECT mais ne décrit pas sa fonction.

La méthode CONNECT demande à une passerelle de tunnel de créer une connexion TCP vers un serveur et un port de destination arbitraires et de relayer aveuglément les données ultérieures entre le client et le serveur.

La figure 8-10 montre comment fonctionne la méthode CONNECT pour établir un tunnel vers une passerelle :

• Dans la Figure 8-10a, le client envoie une requête CONNECT à la passerelle du tunnel. La méthode CONNECT du client demande à la passerelle du tunnel d'ouvrir une connexion TCP (ici, vers l'hôte nommé orders.joes-hardware.com sur le port 443, le port SSL normal).

• La connexion TCP est créée dans la Figure 8-10b et la Figure 8-10c.

• Une fois la connexion TCP établie, la passerelle notifie le client

(Figure 8-10d) en envoyant une réponse HTTP 200 Connection Established.

• À ce stade, le tunnel est configuré. Toutes les données envoyées par le client via le tunnel HTTP seront relayées directement vers la

connexion TCP sortante, et toutes les données envoyées par le serveur seront relayées au client via le tunnel HTTP.

Figure 8-10. Utilisation de CONNECT pour établir un tunnel SSL

L'exemple de la figure 8-10 décrit un tunnel SSL, où le trafic SSL est envoyé via une connexion HTTP, mais la méthode CONNECT peut être utilisée pour établir une connexion TCP vers n'importe quel serveur utilisant n'importe quel protocole.

Demandes CONNECT

La syntaxe de CONNECT est identique à celle des autres méthodes HTTP, à l'exception de la ligne de départ. L'URI de la requête est remplacée par un nom d'hôte, suivi de deux points, puis d'un numéro de port. L'hôte et le port doivent être spécifiés :

CONNECTER home.netscape.com:443 HTTP/1.0

Agent utilisateur : Mozilla/4.0

Après la ligne de départ, il y a zéro ou plusieurs champs d'en-tête de requête HTTP, comme dans les autres messages HTTP. Comme d'habitude, les lignes se terminent par des CRLF et la liste des en-têtes se termine par un

CRLF nu.

Réponses CONNECT

Une fois la requête envoyée, le client attend une réponse de la passerelle. Comme pour les messages HTTP classiques, un code de réponse 200 indique la réussite. Par convention, la raison de la réponse est généralement définie sur « Connexion établie » :

HTTP/1.0 200 Connexion établie

Agent proxy : Netscape-Proxy/1.1

Contrairement aux réponses HTTP classiques, la réponse n'a pas besoin d'inclure d'en-tête ContentType. Aucun type de contenu n'est requis*, car la connexion devient un relais d'octets bruts, et non un transporteur de messages.

Tunneling de données, synchronisation et gestion des connexions

Les données tunnelisées étant opaques pour la passerelle, celle-ci ne peut prédire l'ordre et le flux des paquets. Une fois le tunnel établi, les données peuvent circuler librement dans toutes les directions et à tout moment.†

Pour optimiser les performances, les clients sont autorisés à envoyer des données de tunnel après l'envoi de la requête CONNECT, mais avant la réception de la réponse. Cela permet d'accélérer

l'acheminement des données vers le serveur, mais cela implique que la passerelle soit capable de gérer correctement les données suivant la requête. En particulier, la passerelle ne peut pas supposer qu'une requête d'E/S réseau ne renverra que des données d'en-tête, et elle doit s'assurer de transmettre au serveur toutes les données lues avec l'en-tête, une fois la connexion établie. Clients qui acheminent les données par pipeline

* Les spécifications futures pourraient définir un type de support pour les tunnels (par exemple, application/tunnel), pour des raisons d'uniformité.

† Les deux extrémités du tunnel (le client et la passerelle) doivent être prêtes à accepter les paquets de l'une ou l'autre des connexions à tout moment et doivent transmettre ces données immédiatement. Le protocole tunnelé pouvant inclure des dépendances de données, aucune extrémité du tunnel ne peut ignorer les données d'entrée. Une consommation insuffisante de données à une extrémité du tunnel peut bloquer le producteur à l'autre extrémité, entraînant un blocage.

après la demande, il faut être prêt à renvoyer les données de la demande si la réponse revient sous la forme d'un défi d'authentification ou d'un autre état non 200 et non fatal. *

Si l'un des points de terminaison du tunnel est déconnecté, toutes les données en attente provenant de ce point seront transmises à l'autre, puis cette dernière sera également interrompue par le proxy. Si des données non transmises au point de terminaison en cours de fermeture sont présentes, elles seront supprimées.

Tunneling SSL

Les tunnels web ont été initialement développés pour acheminer le trafic SSL chiffré à travers les pare-feu. De nombreuses organisations acheminent l'ensemble du trafic via des routeurs de filtrage de paquets et des serveurs proxy afin de renforcer la sécurité. Cependant, certains protocoles, comme le SSL chiffré, ne peuvent pas être traités par les serveurs proxy traditionnels, car les informations sont chiffrées. Les tunnels permettent au trafic SSL de traverser le pare-feu HTTP du port 80 via une connexion HTTP (Figure 8-11).

Figure 8-11. Les tunnels permettent au trafic non HTTP de circuler via des connexions HTTP.

Pour permettre au trafic SSL de circuler à travers les pare-feu proxy existants, une fonctionnalité de tunneling a été ajoutée à HTTP, dans laquelle les données brutes et cryptées sont placées dans les messages HTTP et envoyées via les canaux HTTP normaux (Figure 8-12).

* Essayez de ne pas pipeliner plus de données que ce que peut contenir le reste du paquet TCP de la requête. Pipeliner plus de données peut entraîner une réinitialisation TCP du client si la passerelle ferme la connexion avant que tous les paquets TCP pipelinés ne soient reçus. Une réinitialisation TCP peut entraîner la perte de la réponse de la passerelle reçue par le client, l'empêchant ainsi de déterminer si l'échec est dû à une erreur réseau, à un contrôle d'accès ou à une demande d'authentification.

Figure 8-12. Connexion SSL directe vs. connexion SSL tunnelisée

Dans la figure 8-12a, le trafic SSL est envoyé directement à un serveur web sécurisé (sur le port SSL 443). Dans la figure 8-12b, le trafic SSL est encapsulé dans des messages HTTP et transmis via des connexions HTTP sur le port 80, jusqu'à ce qu'il soit décapsulé et converti en connexions SSL normales.

Les tunnels sont souvent utilisés pour laisser passer le trafic non HTTP à travers les pare-feu filtrant les ports. Cela peut être utile, par exemple, pour permettre au trafic SSL sécurisé de traverser les pare-feu. Cependant, cette fonctionnalité peut être détournée, permettant à des protocoles malveillants de pénétrer dans une organisation via le tunnel HTTP.

Tunneling SSL versus passerelles HTTP/HTTPS

Le protocole HTTPS (HTTP sur SSL) peut également être utilisé comme passerelle, comme les autres protocoles : la passerelle (au lieu du client) initie la session SSL avec le serveur HTTPS distant, puis effectue la transaction HTTPS pour le client. La réponse est reçue et déchiffrée par le proxy, puis envoyée au client via HTTP (non sécurisé). C'est ainsi que les passerelles gèrent le FTP. Cependant, cette approche présente plusieurs inconvénients :

• La connexion client-passerelle est un HTTP normal et non sécurisé.

• Le client n'est pas en mesure d'effectuer l'authentification client SSL (authentification basée sur des certificats X509) sur le serveur distant, car le proxy est la partie authentifiée.

• La passerelle doit prendre en charge une implémentation SSL complète.

Notez que ce mécanisme, s'il est utilisé pour le tunneling SSL, ne nécessite pas l'implémentation de SSL dans le proxy. La session SSL est établie entre le client générant la requête et le serveur web de destination (sécurisé) ; le serveur proxy intermédiaire se contente de tunneliser les données chiffrées et n'intervient pas dans la transaction sécurisée.

Authentification du tunnel

D'autres fonctionnalités de HTTP peuvent être utilisées avec les tunnels, le cas échéant. En particulier, l'authentification proxy peut être utilisée avec les tunnels pour authentifier le droit d'un client à utiliser un tunnel (figure 8-13).

Figure 8-13. Les passerelles peuvent authentifier un client par proxy avant de l'autoriser à utiliser un tunnel.

Considérations sur la sécurité des tunnels

En général, la passerelle tunnel ne peut pas vérifier que le protocole utilisé est bien celui qu'elle est censée utiliser. Ainsi, par exemple, des utilisateurs malveillants peuvent utiliser des tunnels SSL pour acheminer le trafic de jeux Internet à travers un pare-feu d'entreprise, ou des utilisateurs malveillants peuvent utiliser des tunnels pour ouvrir des sessions Telnet ou envoyer des e-mails contournant les scanners de messagerie de l'entreprise.

Pour minimiser l'abus des tunnels, la passerelle doit ouvrir des tunnels uniquement pour des ports particuliers bien connus, tels que 443 pour HTTPS.

Relais

Les relais HTTP sont de simples proxys HTTP qui ne respectent pas entièrement les spécifications HTTP. Ils traitent suffisamment de données HTTP pour établir des connexions, puis transmettent aveuglément les octets.

HTTP étant complexe, il est parfois utile d'implémenter des proxys simples qui se contentent de transférer le trafic à l'aveugle, sans exécuter toute la logique des en-têtes et des méthodes. Les relais aveugles étant faciles à implémenter, ils sont parfois utilisés pour fournir un filtrage, des diagnostics ou une transformation de contenu simples. Cependant, leur déploiement doit être effectué avec la plus grande prudence, en raison du risque important de problèmes d'interopérabilité.

L'un des problèmes les plus courants (et notoires) rencontrés avec certaines implémentations de relais aveugles simples est leur risque de bloquer les connexions persistantes, car elles ne traitent pas correctement l'en-tête de connexion. Cette situation est illustrée à la figure 8-14.

Figure 8-14. Les relais aveugles simples peuvent se bloquer s'ils sont monotâches et ne prennent pas en charge l'en-tête de connexion.

Voici ce qui se passe dans cette figure :

• Dans la figure 8-14a, un client Web envoie un message au relais, incluant l'en-tête Connection: Keep-Alive, demandant une connexion maintenue si possible. Le client attend une réponse pour savoir si sa demande de canal maintenu a été acceptée.

• Le relais reçoit la requête HTTP, mais ne comprend pas l'en-tête de connexion ; il transmet donc le message mot pour mot au serveur (Figure 8-14b). Cependant, l'en-tête de connexion est un en-tête saut par saut ; il

Cela ne s'applique qu'à un seul lien de transport et ne devrait pas être transmis à toute la chaîne. De mauvaises choses sont sur le point de se produire !

• Dans la figure 8-14b, la requête HTTP relayée arrive au serveur web. Lorsque le serveur web reçoit l'en-tête proxy « Connection: Keep-Alive », il conclut à tort que le relais (qui ressemble à n'importe quel autre client pour le serveur) souhaite communiquer en mode « keep-alive » ! Le serveur web accepte et renvoie un en-tête de réponse « Connection: Keep-Alive » (voir la figure 8-14c). À ce stade, le serveur web pense communiquer en mode « keep-alive » avec le relais et s'en conformera aux règles. Mais le relais ignore tout de cette fonctionnalité.

• Dans la figure 8-14d, le relais renvoie le message de réponse du serveur web au client, en lui transmettant l'en-tête « Connexion : Keep-Alive ». Le client voit cet en-tête et suppose que le relais a accepté de communiquer avec le serveur. À ce stade, le client et le serveur pensent communiquer avec le serveur, mais le relais auquel ils communiquent ignore tout de cette communication.

• Comme le relais ignore tout du maintien de la connexion active, il renvoie toutes les données reçues au client, en attendant que le serveur d'origine ferme la connexion. Or, le serveur d'origine refuse de fermer la connexion, pensant que le relais lui a demandé de la maintenir ouverte ! Le relais reste donc bloqué en attendant la fermeture de la connexion.

• Lorsque le client reçoit le message de réponse de la figure 8-14d, il passe directement à la requête suivante, en envoyant une autre au relais sur la connexion maintenue (figure 8-14e). Les relais simples n'attendent généralement jamais une autre requête sur la même connexion. Le navigateur tourne sans avancer.

Il existe des moyens de rendre les relais légèrement plus intelligents et d'éliminer ces risques, mais toute simplification des proxys comporte un risque d'interopérabilité. Si vous créez des relais HTTP simples pour un usage particulier, soyez prudent quant à leur utilisation. Pour tout

déploiement à grande échelle, il est fortement conseillé d'utiliser un véritable serveur proxy compatible HTTP.

Pour plus d'informations sur les relais et la gestion des connexions, voir « Keep-Alive et Dumb Proxies » au chapitre 4.

Pour plus d'informations

Pour plus d'informations, reportez-vous à :

http://www.w3.org/Protocols/rfc2616/rfc2616.txt

RFC 2616, « Protocole de transfert hypertexte », par R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Mastinter, P. Leach et T. Berners-Lee.

Serveurs proxy Web

Ari Luotonen, Livres informatiques de Prentice Hall.

Pour plus d'informations

http://www.alternic.org/drafts/drafts-lm/draft-luotonen-web-proxy-tunneling-01.txt

« Tunneling des protocoles basés sur TCP via des serveurs proxy Web », par Ari Luotonen.

http://cgi-spec.golux.com

Page du projet RFC sur l'interface de passerelle commune.

http://www.w3.org/TR/2001/WD-soap12-part0-20011217/ W3C — Projet de travail SOAP version 1.2.

Programmation de services Web avec SOAP

James Snell, Doug Tidwell et Pavel Kulchenko, O'Reilly & Associates, Inc.

http://www.w3.org/TR/2002/WD-wsa-reqs-20020429

W3C — Exigences relatives à l'architecture des services Web.

Notions essentielles sur les services Web

Ethan Cermai, O'Reilly & Associates, Inc.

Chapitre 9 Agent utilisateur Ceci est le titre du livre CHAPITRE 9

Robots Web

Nous continuons notre visite de l'architecture HTTP avec un examen attentif des agents utilisateurs auto-animés appelés robots Web.

Les robots web sont des logiciels qui automatisent une série de transactions web sans intervention humaine. De nombreux robots errent de site web en site, récupérant du contenu, suivant des hyperliens et traitant les données trouvées. Ces robots portent des

noms savants tels que « crawlers », « spiders », « worms » et « bots » en raison de leur façon d'explorer automatiquement les sites web, apparemment dotés de leur propre intelligence.

Voici quelques exemples de robots Web :

• Les robots de graphiques boursiers envoient des requêtes HTTP GET aux serveurs du marché boursier toutes les quelques minutes et utilisent les données pour créer des graphiques de tendance des cours des actions.

• Les robots de recensement Web collectent des informations de « recensement » sur l'échelle et l'évolution du World Wide Web. Ils parcourent le Web en comptant le nombre de pages et

enregistrement de la taille, de la langue et du type de support de chaque page.*

• Les robots des moteurs de recherche collectent tous les documents qu'ils trouvent pour créer des bases de données de recherche.

• Les robots de comparaison de prix collectent des pages Web à partir de catalogues de magasins en ligne pour créer des bases de données de produits et de leurs prix.

Les rampants et le rampant

Les robots d'indexation sont des robots qui parcourent récursivement les réseaux d'information, récupérant d'abord une page web, puis toutes les pages web vers lesquelles elle pointe, puis toutes les pages web vers lesquelles ces pages pointent, et ainsi de suite. Lorsqu'un robot suit récursivement des liens web, on l'appelle un robot d'indexation ou une araignée, car il « rampe » le long du réseau créé par HTML.

hyperliens.

* http://www.netcraft.com collecte d'excellentes mesures de recensement sur les types de serveurs utilisés par les sites sur le Web.

Les moteurs de recherche utilisent des robots d'indexation pour parcourir le Web et récupérer tous les documents qu'ils rencontrent. Ces documents sont ensuite traités pour créer une base de données consultable, permettant aux utilisateurs de trouver des documents contenant des mots clés. Avec des milliards de pages web à consulter et à récupérer, ces robots d'indexation sont forcément parmi les plus sophistiqués. Examinons plus en détail leur fonctionnement.

Par où commencer : le « Root Set »

Avant de libérer votre robot d'exploration, vous devez lui donner un point de départ. L'ensemble initial d'URL qu'un robot visite est appelé

ensemble racine. Pour choisir cet ensemble, vous devez sélectionner des URL provenant d'emplacements suffisamment différents pour que l'exploration de tous les liens vous mène finalement à la plupart des pages web qui vous intéressent.

Quel est le bon ensemble de racines à utiliser pour explorer le Web illustré à la figure 9-1 ? Comme sur le Web réel, il n'existe pas de document unique qui renvoie à tous les documents. Si vous commencez par le document A de la figure 9-1, vous pouvez accéder aux documents B, C et D, puis E et F, puis J, puis K. Mais il n'existe pas de chaîne de liens de A à G ou de A à N.

Figure 9-1. Un ensemble de racines est nécessaire pour accéder à toutes les pages.

Certaines pages web de ce réseau, comme S, T et U, sont quasiment abandonnées, isolées, sans aucun lien pointant vers elles. Peut-être ces pages isolées sont-elles récentes et inconnues. Ou peut-être sont-elles vraiment anciennes ou obscures.

En général, il n'est pas nécessaire d'avoir trop de pages dans l'ensemble racine pour couvrir une grande partie du Web. Dans la figure 9-1, seuls les éléments A, G et S sont nécessaires pour atteindre toutes les pages.

En règle générale, un bon ensemble racine comprend les sites web importants et populaires (par exemple, http://www.yahoo.com), une liste des pages nouvellement créées et une liste des pages peu connues auxquelles il n'est pas souvent fait référence. De nombreux robots d'indexation de production à grande échelle, comme ceux utilisés par les moteurs de recherche, permettent aux utilisateurs d'ajouter des pages nouvelles ou peu connues à l'ensemble racine.

Cet ensemble de racines se développe au fil du temps et constitue la liste de semences pour toutes les nouvelles pousses.

Extraction de liens et normalisation des liens relatifs

Lorsqu'un robot d'exploration parcourt le Web, il récupère constamment des pages HTML. Il doit analyser les liens URL de chaque page récupérée et les ajouter à la liste des pages à explorer. Au cours de l'exploration, cette liste s'allonge souvent rapidement, car le robot découvre de nouveaux liens à explorer*. Les robots d'exploration doivent effectuer une analyse HTML simple pour extraire ces liens et convertir les URL relatives en URL absolues. La section « URL relatives » du chapitre 2 explique comment effectuer cette conversion.

Évitement du vélo

Lorsqu'un robot explore une toile, il doit faire très attention à ne pas se retrouver bloqué dans une boucle. Observez le robot de la figure 9-2 :

• Dans la figure 9-2a, le robot récupère la page A, voit que A est lié à B et récupère la page B.

• Dans la figure 9-2b, le robot récupère la page B, voit que B est lié à C et récupère la page C.

• Dans la figure 9-2c, le robot récupère la page C et voit que C est lié à A. Si le robot récupère à nouveau la page A, il se retrouvera dans un cycle, récupérant A, B, C, A, B, C, A...

Figure 9-2. Explorer un réseau d'hyperliens

Les robots doivent savoir où ils sont passés pour éviter les cycles. Ces derniers peuvent entraîner des pièges qui peuvent stopper ou ralentir leur progression.

Boucles et Dups

Les cycles sont mauvais pour les robots d'exploration pour au moins trois raisons :

• Ils entraînent le robot dans une boucle où il peut rester coincé. Une boucle peut forcer un robot mal conçu à tourner en rond, passant tout son temps à chercher.

* Dans « Évitement des cycles », nous commençons à aborder la nécessité pour les robots d'exploration de se souvenir de leurs passages. Lors d'une exploration, la liste des URL découvertes s'allonge jusqu'à ce que l'espace web soit exploré en profondeur et que le robot atteigne un point où il ne découvre plus de nouveaux liens.

Les mêmes pages réapparaissent sans cesse. Le robot peut consommer une grande quantité de bande passante réseau et être incapable d'accéder à d'autres pages.

• Pendant que le robot d'exploration récupère les mêmes pages de manière répétée, le serveur web de l'autre côté est saturé. Si le robot d'exploration est bien connecté, il peut surcharger le site web et empêcher tout utilisateur réel d'y accéder. Un tel déni de service peut donner lieu à des poursuites judiciaires.

• Même si la boucle n'est pas un problème en soi, le robot d'indexation récupère un grand nombre de pages dupliquées (souvent appelées « doublons », synonyme de « boucles »). L'application du robot d'indexation sera inondée de contenu dupliqué, ce qui peut la rendre inutilisable. Un moteur de recherche Internet qui renvoie des centaines de résultats pour une même page en est un exemple.

Sentiers de miettes de pain

Malheureusement, il n'est pas toujours facile de savoir où l'on est allé. Au moment où j'écris ces lignes, il existe des milliards de pages web

distinctes sur Internet, sans compter le contenu généré par les passerelles dynamiques.

Si vous souhaitez explorer une grande partie du contenu web mondial, vous devez vous préparer à visiter des milliards d'URL. Suivre les URL visitées peut s'avérer complexe. Face à ce nombre important d'URL, vous devez utiliser des structures de données sophistiquées pour identifier rapidement celles que vous avez visitées. Ces structures doivent être efficaces en termes de vitesse et d'utilisation de la mémoire.

La rapidité est un facteur important, car des centaines de millions d'URL nécessitent des structures de recherche rapides. Une recherche exhaustive dans des listes d'URL est hors de question. Un robot devra au minimum utiliser un arbre de recherche ou une table de hachage pour déterminer rapidement si une URL a été visitée.

Des centaines de millions d'URL occupent également beaucoup d'espace. Si une URL moyenne comporte 40 caractères et qu'un robot web explore 500 millions d'URL (soit une petite partie du Web), une structure de données de recherche pourrait nécessiter 20 Go ou plus de mémoire rien que pour stocker les URL (40 octets par URL × 500 millions d'URL = 20 Go) !

Voici quelques techniques utiles que les robots d’exploration Web à grande échelle utilisent pour gérer les endroits qu’ils visitent :

Arbres et tables de hachage

Les robots sophistiqués peuvent utiliser un arbre de recherche ou une table de hachage pour suivre les URL visitées. Ces structures de données logicielles accélèrent considérablement la recherche d'URL.

Cartes de bits de présence avec perte

Pour minimiser l'espace, certains robots d'exploration à grande échelle utilisent des structures de données avec perte, telles que des tableaux de bits de présence. Chaque URL est convertie en un nombre de taille fixe par une fonction de hachage, auquel est associé un « bit de présence » dans un tableau. Lorsqu'une URL est explorée, le bit de présence correspondant est défini. Si le bit de présence est déjà défini, le robot d'exploration considère que l'URL a déjà été explorée.*

Points de contrôle

Assurez-vous de sauvegarder la liste des URL visitées sur le disque, au cas où le programme robot planterait.

Partitionnement

Avec le développement du Web, il peut devenir difficile d'effectuer une exploration avec un seul robot sur un seul ordinateur. Cet ordinateur

peut manquer de mémoire, d'espace disque, de puissance de calcul ou de bande passante réseau pour effectuer une exploration.

Certains robots web à grande échelle utilisent des « fermes » de robots, chacun étant un ordinateur distinct, travaillant en tandem. Chaque robot se voit attribuer une « tranche » d'URL spécifique, dont il est responsable. Ensemble, les robots explorent le Web. Chaque robot peut avoir besoin de communiquer pour échanger des URL, pour pallier les dysfonctionnements de ses homologues ou pour coordonner ses efforts.

Un bon ouvrage de référence pour la mise en œuvre de structures de données volumineuses est « Gestion des gigaoctets : compression et indexation de documents et d'images », de Witten et al. (Morgan Kaufmann). Cet ouvrage regorge d'astuces et de techniques pour gérer de grandes quantités de données.

Alias et cycles de robot

Même avec les bonnes structures de données, il est parfois difficile de savoir si vous avez déjà visité une page, en raison de l'« aliasing » des URL. Deux URL sont des alias si elles semblent différentes, mais renvoient en réalité à la même ressource.

Le tableau 9-1 illustre quelques manières simples par lesquelles différentes URL peuvent pointer vers la même ressource.

Tableau 9-1. Différentes URL pointant vers les mêmes documents

Première URL Deuxième URL Lorsqu'il est aliasé

a http://www.foo.com/bar.html http://www.foo.com:80/bar.html Le port est 80 par défaut

b http://www.foo.com/~fred http://www.foo.com/%7Ffred %7F est identique à ~

c http://www.foo.com/x.html#early http://www.foo.com/x.html#middle Les balises ne changent pas la page

d http://www.foo.com/readme.htm http://www.foo.com/README.HTM Serveur insensible à la casse

e http://www.foo.com/ http://www.foo.com/index.html La page par défaut est index.html

f http://www.foo.com/index.html http://209.231.87.45/index.html www.foo.com a cette adresse IP

* Étant donné le nombre potentiellement infini d'URL et le nombre limité de bits dans le tableau de bits de présence, il existe un risque de collision : deux URL peuvent correspondre au même bit de présence. Dans ce cas, le robot d'exploration conclut à tort qu'une page a été

explorée alors que ce n'est pas le cas. En pratique, l'utilisation d'un nombre élevé de bits de présence peut rendre cette situation très improbable. La conséquence d'une collision est l'omission d'une page lors de l'exploration.

URL canoniques

La plupart des robots web tentent d'éliminer d'emblée les alias évidents en « canonisant » les URL pour les rendre standardisées. Un robot peut d'abord convertir chaque URL en une forme canonique, en :

1. Ajout de « :80 » au nom d'hôte, si le port n'est pas spécifié

2. Conversion de tous les caractères échappés %xx en leurs équivalents caractères

3. Suppression des balises #

Ces étapes peuvent éliminer les problèmes d'aliasing présentés dans le tableau 9-1a–c. Cependant, sans informations sur le serveur web concerné, le robot n'a aucun moyen efficace d'éviter les doublons du tableau 9-1d–f :

• Le robot devrait savoir si le serveur Web est insensible à la casse pour éviter l'alias du tableau 9-1d.

• Le robot aurait besoin de connaître la configuration de la page d'index du serveur Web pour ce répertoire pour savoir si les URL du tableau 9-1e étaient des alias.

• Le robot aurait besoin de savoir si le serveur Web était configuré pour effectuer un hébergement virtuel (abordé au chapitre 5) pour savoir si les URL du tableau 9-1f étaient des alias, même s'il savait que le nom d'hôte et l'adresse IP faisaient référence au même ordinateur physique.

La canonicalisation d'URL peut éliminer les alias syntaxiques de base, mais les robots rencontreront d'autres alias d'URL qui ne peuvent pas être éliminés en convertissant les URL en formes standard.

Cycles de liaison du système de fichiers

Les liens symboliques sur un système de fichiers peuvent provoquer un cycle particulièrement insidieux, car ils peuvent créer l'illusion d'une hiérarchie de répertoires infiniment profonde là où il n'en existe pas. Ces cycles de liens symboliques résultent généralement d'une erreur involontaire de l'administrateur du serveur, mais peuvent également être créés par des webmasters malveillants, servant de piège aux robots.

La figure 9-3 illustre deux systèmes de fichiers. Dans la figure 9-3a, subdir est un répertoire normal. Dans la figure 9-3b, subdir est un lien symbolique pointant vers /. Dans les deux figures, supposons que le fichier /index.html contienne un lien hypertexte vers le fichier subdir/index.html.

En utilisant le système de fichiers de la figure 9-3a, un robot d'exploration Web peut effectuer les actions suivantes :

1. OBTENEZ http://www.foo.com/index.html

Obtenez /index.html, recherchez le lien vers subdir/index.html.

2. OBTENEZ http://www.foo.com/subdir/index.html

Obtenez subdir/index.html, recherchez le lien vers subdir/logo.gif.

3. OBTENEZ http://www.foo.com/subdir/logo.gif

Obtenez subdir/logo.gif, plus de liens, c'est fait.

Figure 9-3. Cycles de liens symboliques

Mais dans le système de fichiers de la figure 9-3b, ce qui suit peut se produire :

1. OBTENEZ http://www.foo.com/index.html

Obtenez /index.html, recherchez le lien vers subdir/index.html.

2. OBTENEZ http://www.foo.com/subdir/index.html

Obtenez subdir/index.html, mais récupérez le même index.html.

3. OBTENEZ http://www.foo.com/subdir/subdir/index.html Obtenez subdir/subdir/index.html.

4. OBTENEZ http://www.foo.com/subdir/subdir/subdir/index.html Obtenez subdir/subdir/subdir/index.html.

Le problème avec la figure 9-3b est que subdir/ est un cycle de retour à /, mais comme les URL sont différentes, le robot ne sait pas, à partir de l'URL seule, que les documents sont identiques. Le robot, sans méfiance, court le risque de tomber dans une boucle. Sans détection de boucle, ce cycle se poursuit, souvent jusqu'à ce que la longueur de l'URL dépasse les limites du robot ou du serveur.

Espaces Web virtuels dynamiques

Il est possible pour des webmasters malveillants de créer intentionnellement des boucles d'exploration sophistiquées pour piéger des robots innocents et sans méfiance. Il est notamment facile de publier une URL qui ressemble à un fichier normal, mais qui est en réalité une application passerelle. Cette application peut générer instantanément du code HTML contenant des liens vers des URL imaginaires sur le même serveur. Lorsque ces URL imaginaires sont demandées, le serveur malveillant crée une nouvelle page HTML avec de nouvelles URL imaginaires.

Le serveur web malveillant peut entraîner le pauvre robot dans un voyage digne d'Alice au pays des merveilles à travers un espace web virtuel infini, même s'il ne contient aucun fichier. Pire encore, il peut rendre la détection du cycle très difficile pour le robot, car les URL et le code HTML peuvent être très différents à chaque fois. La figure 9-4 montre un exemple de serveur web malveillant générant du contenu fictif.

Figure 9-4. Exemple d'espace Web dynamique malveillant

Plus souvent, des webmasters bien intentionnés créent involontairement un piège à robots d'indexation via des liens symboliques ou du contenu dynamique. Prenons l'exemple d'un programme de calendrier CGI générant un calendrier mensuel et un lien vers le mois suivant. Un utilisateur réel ne demanderait pas indéfiniment le lien vers le mois suivant, mais un robot ignorant la nature dynamique du contenu pourrait continuer à demander ces ressources.

indéfiniment.*

Éviter les boucles et les doublons

Il n'existe pas de méthode infaillible pour éviter tous les cycles. En pratique, les robots bien conçus doivent intégrer un ensemble d'heuristiques pour tenter d'éviter les cycles.

En règle générale, plus un robot est autonome (moins de surveillance humaine), plus il risque d'être en difficulté. Les développeurs de robots doivent faire un compromis : ces heuristiques peuvent aider à éviter les problèmes, mais elles sont aussi quelque peu déroutantes.

« avec perte », car vous risquez de passer à côté de contenus valides qui semblent suspects.

* Ceci est un exemple réel mentionné sur http://www.searchtools.com/robots/robot-checklist.html pour le site de calendrier http://cgi.umbc.edu/cgi-bin/WebEvent/webevent.cgi. En raison de ce type de contenu dynamique, de nombreux robots refusent d'explorer les pages dont l'URL contient la sous-chaîne « cgi ».

Voici quelques techniques que les robots utilisent pour mieux se comporter dans un réseau rempli de dangers robotiques :

URL canoniques

Évitez les alias syntaxiques en convertissant les URL en forme standard.

Ramper en largeur d'abord

Les robots d'exploration disposent d'un large éventail d'URL potentielles à explorer simultanément. En planifiant les visites d'URL en

largeur, sur plusieurs sites web, vous pouvez minimiser l'impact des cycles. Même en cas de piège robot, vous pouvez toujours récupérer des centaines de milliers de pages d'autres sites web avant de revenir chercher une page du cycle. En explorant en profondeur un seul site, vous risquez de tomber sur un cycle sans jamais pouvoir vous en sortir.*

Étranglement†

Limitez le nombre de pages que le robot peut récupérer sur un site web sur une période donnée. Si le robot entre en cycle et tente continuellement d'accéder aux alias d'un site, vous pouvez limiter le nombre total de doublons générés et le nombre total d'accès au serveur en limitant le nombre de pages.

Limiter la taille de l'URL

Le robot peut refuser d'explorer les URL au-delà d'une certaine longueur (1 Ko est courant). Si un cycle entraîne une augmentation de la taille de l'URL, une limite de longueur finira par interrompre le cycle. Certains serveurs web échouent lorsqu'ils reçoivent des URL longues, et les robots pris dans un cycle d'augmentation d'URL peuvent provoquer le plantage de certains serveurs web. Cela peut amener les webmasters à interpréter à tort le robot comme un attaquant par déni de service.

Par précaution, cette technique peut certainement conduire à manquer du contenu. De nombreux sites utilisent aujourd'hui les URL pour gérer l'état des utilisateurs (par exemple, en stockant les identifiants des utilisateurs dans les URL référencées dans une page). La taille des URL peut être un moyen délicat de limiter l'exploration ; cependant, elle peut fournir un excellent indicateur permettant à l'utilisateur de vérifier ce qui se passe sur un site particulier, en signalant une erreur lorsque les URL demandées atteignent une certaine taille.

Liste noire d'URL/site

Tenez une liste des sites et URL connus correspondant aux cycles et pièges des robots, et évitez-les comme la peste. Dès que de nouveaux problèmes sont détectés, ajoutez-les à la liste noire.

Cela nécessite une intervention humaine. Cependant, la plupart des robots d'exploration à grande échelle en production disposent aujourd'hui d'une liste noire, utilisée pour éviter certains sites en raison de problèmes inhérents ou d'éléments malveillants. Cette liste noire peut également servir à éviter certains sites qui ont fait grand bruit après avoir été explorés.‡

* L'exploration en largeur est généralement une bonne idée, car elle permet de répartir les requêtes de manière plus homogène et de ne pas surcharger un serveur. Cela permet de minimiser l'utilisation des ressources par un robot sur un serveur.

† La limitation du taux de requêtes est également abordée dans « Étiquette des robots ».

‡ « Exclure les robots » explique comment les sites peuvent éviter d'être explorés, mais certains utilisateurs refusent d'utiliser ce simple mécanisme de contrôle et deviennent très furieux lorsque leurs sites sont explorés.

Détection de motifs

Les cycles causés par les liens symboliques du système de fichiers et autres erreurs de configuration similaires ont tendance à suivre des schémas ; par exemple, l'URL peut croître avec des composants dupliqués. Certains robots considèrent les URL contenant des composants répétés comme des cycles potentiels et refusent d'explorer celles contenant plus de deux ou trois composants répétés.

Toutes les répétitions ne sont pas immédiates (par exemple, « /subdir/subdir/subdir... »). Il est possible d'avoir des cycles de période 2 ou d'autres intervalles, comme « /subdir/images/subdir/images/subdir/images/... ». Certains robots recherchent des motifs répétitifs de plusieurs périodes différentes.

Empreinte digitale du contenu

L'empreinte digitale est une méthode plus directe de détection des doublons, utilisée par certains robots d'exploration web sophistiqués. Les robots utilisant l'empreinte digitale de contenu extraient les octets du contenu de la page et calculent une somme de contrôle. Cette somme de contrôle est une représentation compacte du contenu de la page. Si un robot récupère une page dont il a déjà vu la somme de contrôle, les liens de cette page ne sont pas explorés. Si le robot a déjà vu le contenu de la page, il a déjà lancé l'exploration des liens.

La fonction de somme de contrôle doit être choisie de manière à réduire au minimum la probabilité que deux pages différentes aient la même somme de contrôle. Les fonctions de résumé de message telles que MD5 sont courantes pour l'empreinte digitale.

Étant donné que certains serveurs web modifient dynamiquement les pages à la volée, les robots omettent parfois certaines parties du contenu des pages, comme les liens intégrés, du calcul de la somme de contrôle. Cependant, les inclusions dynamiques côté serveur qui personnalisent le contenu arbitraire des pages (ajout de dates, de compteurs d'accès, etc.) peuvent empêcher la détection des doublons.

Surveillance humaine

Le Web est un monde sauvage. Votre courageux robot finira par rencontrer un problème qu'aucune de vos techniques ne pourra détecter. Tous les robots de production doivent être conçus avec des fonctions de diagnostic et de journalisation, afin que les humains

puissent facilement suivre leur progression et être avertis rapidement en cas d'anomalie. Il arrive que des internautes en colère vous signalent le problème en vous envoyant des courriels malveillants.

Les bonnes heuristiques d'exploration pour explorer des ensembles de données aussi vastes que le Web sont en constante évolution. Les règles sont élaborées au fil du temps et adaptées à mesure que de nouveaux types de ressources sont ajoutés au Web. Les bonnes règles évoluent constamment.

De nombreux robots d'exploration plus petits et plus personnalisés contournent certains de ces problèmes, car les ressources (serveurs, bande passante réseau, etc.) affectées par un robot d'exploration défaillant sont gérables, voire sous le contrôle de la personne effectuant l'exploration (comme sur un site intranet). Ces robots d'exploration s'appuient davantage sur une surveillance humaine pour prévenir les problèmes.

HTTP robotique

Les robots ne sont pas différents des autres programmes clients HTTP. Ils doivent eux aussi respecter les règles de la spécification HTTP. Un robot qui effectue des requêtes HTTP et se présente comme client HTTP/1.1 doit utiliser les en-têtes de requête HTTP appropriés.

De nombreux robots tentent d'implémenter le minimum de HTTP nécessaire pour demander le contenu recherché. Cela peut entraîner des problèmes ; cependant, il est peu probable que ce comportement change de sitôt. Par conséquent, de nombreux robots effectuent des requêtes HTTP/1.0, car ce protocole a peu d'exigences.

Identification des en-têtes de requête

Malgré la faible quantité de HTTP prise en charge par les robots, la plupart implémentent et envoient des en-têtes d'identification, notamment l'en-tête HTTP User-Agent. Il est recommandé aux développeurs de robots d'envoyer des informations d'en-tête de base pour informer le site des capacités du robot, de son identité et de son origine.

Ces informations sont utiles pour retrouver le propriétaire d'un robot d'indexation errant et pour fournir au serveur des informations sur les types de contenu que le robot peut traiter. Voici quelques-uns des en-têtes d'identification de base que les développeurs de robots sont encouragés à utiliser :

Agent utilisateur

Indique au serveur le nom du robot effectuant la requête.

Depuis

Fournit l'adresse e-mail de l'utilisateur/administrateur du robot.*

Accepter

Indique au serveur quels types de médias peuvent être envoyés.† Cela peut aider à garantir que le robot ne reçoit que le contenu qui l'intéresse (texte, images, etc.).

Référent

Fournit l'URL du document qui contient l'URL de la demande actuelle.‡

Hébergement virtuel

Les développeurs de robots doivent prendre en charge l'en-tête d'hôte. Compte tenu de la prévalence de l'hébergement virtuel (le chapitre 5 aborde plus en détail les serveurs hébergés virtuellement), l'exclusion de l'en-tête d'hôte est essentielle.

* Un format d'adresse e-mail RFC 822.

† « Accept headers » au chapitre 3 répertorie tous les en-têtes d'acceptation ; les robots peuvent trouver utile d'envoyer des en-têtes tels que Accept-Charset s'ils sont intéressés par des versions particulières.

‡ Cela peut être très utile aux administrateurs de sites qui tentent de déterminer comment un robot a trouvé des liens vers le contenu de leurs sites.

HTTP robotique

L'en-tête HTTP de l'hôte dans les requêtes peut amener les robots à identifier un contenu erroné avec une URL particulière. HTTP/1.1 requiert donc l'utilisation de l'en-tête Host.

La plupart des serveurs sont configurés par défaut pour desservir un site spécifique. Ainsi, un robot d'indexation n'incluant pas l'en-tête Host peut adresser une requête à un serveur desservant deux sites, comme ceux de la figure 9-5 (www.joes-hardware.com et www.foo.com). Si le serveur est configuré pour desservir www.joes-hardware.com par défaut (et ne nécessite pas l'en-tête Host), une requête pour une page sur www.foo.com peut amener le robot d'indexation à récupérer le contenu du site de Joe's Hardware. Pire encore, le robot d'indexation pensera que le contenu de Joe's Hardware provient de www.foo.com. Vous pouvez sans doute imaginer des situations plus fâcheuses encore si des documents provenant de deux sites aux opinions politiques ou autres polémiques étaient diffusés depuis le même serveur.

Figure 9-5. Exemple de docroots virtuels posant problème si aucun en-tête d'hôte n'est envoyé avec la requête.

Demandes conditionnelles

Étant donné l'ampleur de certaines tâches robotiques, il est souvent judicieux de minimiser la quantité de contenu récupérée par un robot. Comme pour les robots des moteurs de recherche, qui peuvent télécharger des milliards de pages web, il est judicieux de ne récupérer le contenu que s'il a été modifié.

Certains de ces robots implémentent des requêtes HTTP conditionnelles*, comparant les horodatages ou les balises d'entité pour vérifier si la dernière version récupérée a été mise à jour. Ce principe est très similaire à celui utilisé par un cache HTTP pour vérifier la validité de la copie locale d'une ressource précédemment récupérée. Consultez le chapitre 7 pour en savoir plus sur la validation des copies locales des ressources par les caches.

* « En-têtes de requête conditionnels » au chapitre 3 donne une liste complète des en-têtes conditionnels qu'un robot peut implémenter.

Gestion des réponses

Comme de nombreux robots cherchent principalement à obtenir le contenu demandé via de simples méthodes GET, ils gèrent rarement les réponses. Cependant, les robots qui utilisent certaines fonctionnalités HTTP (comme les requêtes conditionnelles), ainsi que ceux qui souhaitent mieux explorer et interagir avec les serveurs, doivent être capables de gérer différents types de réponses HTTP.

Codes d'état

En général, les robots doivent être capables de gérer au moins les codes d'état courants ou attendus. Tous les robots doivent comprendre les codes d'état HTTP tels que 200 OK et 404 Not Found. Ils doivent également être capables de gérer les codes d'état qu'ils ne comprennent pas explicitement, en fonction de la catégorie générale de la réponse. Le tableau 3-2 du chapitre 3 détaille les différentes catégories de codes d'état et leur signification.

Il est important de noter que certains serveurs ne renvoient pas toujours les codes d'erreur appropriés. Certains serveurs renvoient même des codes d'état HTTP 200 OK avec le corps du message décrivant une erreur ! Il est difficile d'y remédier ; il s'agit simplement d'un point que les développeurs doivent prendre en compte.

Entités

Outre les informations intégrées aux en-têtes HTTP, les robots peuvent rechercher des informations dans l'entité elle-même. Les balises méta HTML*, telles que la balise méta http-equiv, permettent aux auteurs de contenu d'intégrer des informations supplémentaires sur les ressources. La balise http-equiv elle-même permet aux auteurs de

contenu de remplacer certains en-têtes que le serveur qui gère leur contenu peut afficher :

<meta http-equiv="Actualiser" content="1;URL=index.html">

Cette balise indique au récepteur de traiter le document comme si son en-tête de réponse HTTP contenait un en-tête HTTP Refresh avec la valeur « 1, URL=index.html ».†

Certains serveurs analysent le contenu des pages HTML avant de les envoyer et incluent des directives http-equiv dans leurs en-têtes ; d'autres non. Les développeurs de robots pourraient vouloir analyser les éléments HEAD des documents HTML pour rechercher des directives http-

informations équivalentes. ‡

* « Directives META du robot » répertorie les directives méta supplémentaires que les administrateurs de site et les auteurs de contenu peuvent utiliser pour contrôler le comportement des robots et ce qu'ils font avec les documents récupérés.

† L'en-tête HTTP Refresh est parfois utilisé comme moyen de rediriger les utilisateurs (ou dans ce cas, un robot) d'une page à une autre.

‡ Les balises méta doivent apparaître dans la section HEAD des documents HTML, conformément à la spécification HTML. Cependant, elles apparaissent parfois dans d'autres sections de documents HTML, car tous les documents HTML ne respectent pas la spécification.

HTTP robotique

Ciblage par agent utilisateur

Les administrateurs web doivent garder à l'esprit que de nombreux robots visiteront leurs sites et doivent donc s'attendre à recevoir des requêtes de leur part. De nombreux sites optimisent leur contenu pour différents agents utilisateurs, en s'efforçant de détecter les types de navigateurs afin de garantir la prise en charge de diverses fonctionnalités. Ce faisant, les sites affichent des pages d'erreur plutôt que du contenu aux robots. Une recherche textuelle avec l'expression « votre navigateur ne prend pas en charge les frames » sur certains moteurs de recherche affichera une liste de résultats contenant des pages d'erreur contenant cette expression, alors qu'en réalité, le client HTTP n'était pas un navigateur, mais un robot.

Les administrateurs de site doivent élaborer une stratégie de gestion des requêtes des robots. Par exemple, au lieu de limiter le développement de leur contenu à la prise en charge de navigateurs spécifiques, ils peuvent développer des pages fourre-tout pour les navigateurs et les robots peu riches en fonctionnalités. Au minimum, ils doivent s'attendre à ce que les robots visitent leurs sites sans être pris au dépourvu.

faire.*

Robots mal élevés

Les robots rebelles peuvent semer le chaos de multiples façons. Voici quelques erreurs qu'ils peuvent commettre et leurs conséquences :

Robots en fuite

Les robots émettent des requêtes HTTP beaucoup plus rapidement que les internautes humains et fonctionnent généralement sur des ordinateurs rapides dotés de connexions réseau rapides. Si un robot contient une erreur de programmation ou est pris dans un cycle, il peut imposer une charge importante à un serveur web, potentiellement suffisante pour le surcharger et priver quiconque d'accès. Tous les concepteurs de robots doivent prendre le plus grand soin à intégrer des mesures de protection contre les robots incontrôlables.

URL obsolètes

Certains robots consultent des listes d'URL. Ces listes peuvent être anciennes. Si un site web modifie considérablement son contenu, les robots peuvent demander un grand nombre d'URL inexistantes. Cela agace certains administrateurs de sites web, qui n'apprécient pas que leurs journaux d'erreurs soient remplis de demandes d'accès à des documents inexistants et qui n'apprécient pas que la capacité de leur serveur web soit réduite par la surcharge liée à la gestion des pages d'erreur.

URL longues et erronées

En raison de cycles et d'erreurs de programmation, les robots peuvent demander des URL volumineuses et absurdes aux sites web. Si l'URL est trop longue, elle peut réduire les performances du serveur web, encombrer les journaux d'accès et même provoquer le plantage de serveurs web fragiles.

* « Exclusion des robots » fournit des informations sur la manière dont les administrateurs de site peuvent contrôler le comportement des robots sur leurs sites s'il existe du contenu auquel les robots ne doivent pas accéder.

Robots curieux*

Certains robots peuvent récupérer des URL pointant vers des données privées et les rendre facilement accessibles via les moteurs de recherche et autres applications. Si le propriétaire des données n'a pas activement promu les pages web, il peut considérer la publication robotisée comme une nuisance, au mieux, et une atteinte à la vie privée, au pire.

Cela se produit généralement parce qu'un lien hypertexte vers le contenu « privé » suivi par le robot existe déjà (c'est-à-dire que le contenu n'est pas aussi secret que son propriétaire le pensait, ou que ce dernier a oublié de supprimer un lien hypertexte préexistant). Cela se produit parfois lorsqu'un robot s'acharne à récupérer les documents d'un site, par exemple en récupérant le contenu d'un répertoire, même en l'absence de lien hypertexte explicite.

Les développeurs de robots récupérant de grandes quantités de données sur le Web doivent être conscients que leurs robots sont susceptibles de récupérer des données sensibles à un moment donné, des données que le développeur du site n'a jamais souhaité rendre accessibles sur Internet. Ces données sensibles peuvent inclure des fichiers de mots de passe ou même des informations de carte bancaire. Il est donc essentiel de disposer d'un mécanisme permettant d'ignorer le contenu une fois signalé (et de le supprimer de tout index de recherche ou archive). Les utilisateurs malveillants de moteurs de recherche et d'archives sont connus pour exploiter les capacités des robots d'indexation web à grande échelle pour trouver du contenu. Certains moteurs de recherche, comme Google†, archivent les représentations des pages qu'ils ont explorées. Ainsi, même si le contenu est supprimé, il reste accessible pendant un certain temps.

Accès à la passerelle dynamique

Les robots ne savent pas toujours à quoi ils accèdent. Un robot peut récupérer une URL dont le contenu provient d'une application passerelle. Dans ce cas, les données obtenues peuvent être spécifiques et coûteuses à traiter. De nombreux administrateurs de sites web n'apprécient pas les robots naïfs qui demandent des documents provenant de passerelles.

Hors robots

La communauté des robots a compris les problèmes que pouvait engendrer l'accès robotisé aux sites web. En 1994, une technique simple et volontaire a été proposée pour empêcher les robots d'accéder aux sites qui ne leur conviennent pas et fournir aux webmasters un mécanisme permettant de mieux contrôler leur comportement. Cette norme, baptisée « Norme d'exclusion des robots », est souvent simplement appelée robots.txt, d'après le fichier contenant les informations de contrôle d'accès.

Le principe du fichier robots.txt est simple. Tout serveur web peut fournir un fichier optionnel nommé robots.txt à la racine du document. Ce fichier contient des informations sur les robots autorisés à accéder à telle ou telle partie du serveur. Si un robot suit ce cheminement volontaire,

* En général, si une ressource est disponible sur l'Internet public, elle est probablement référencée quelque part. Peu de ressources sont

véritablement privées, compte tenu du réseau de liens existant sur Internet.

† Consultez les résultats de recherche sur http://www.google.com. Un lien en cache, qui est une copie de la page récupérée et indexée par le robot d'exploration Google, est disponible dans la plupart des résultats.

Par défaut, le robot demande le fichier robots.txt au site web avant d'accéder à toute autre ressource de ce site. Par exemple, le robot de la figure 9-6 souhaite télécharger http://www.joes-hardware.com/specials/acetylene-torches.html depuis Joe's Hardware. Cependant, avant de pouvoir demander la page, il doit vérifier le fichier robots.txt pour savoir s'il est autorisé à la récupérer. Dans cet exemple, le fichier robots.txt ne bloque pas le robot ; il récupère donc la page.

Figure 9-6. Récupération du fichier robots.txt et vérification de l'accessibilité avant l'exploration du fichier cible

La norme d'exclusion des robots

La norme d'exclusion des robots est une norme ad hoc. À l'heure actuelle, aucun organisme officiel de normalisation n'en est propriétaire, et les fournisseurs en implémentent différents sous-ensembles. Néanmoins, une certaine capacité à gérer l'accès des robots aux sites web, même imparfaite, est toujours préférable à l'absence totale de capacité, et la plupart des principaux fournisseurs et robots d'indexation des moteurs de recherche prennent en charge cette norme d'exclusion.

Il existe trois révisions de la norme d'exclusion des robots, bien que la dénomination de ces versions ne soit pas clairement définie. Nous adoptons la numérotation des versions indiquée dans le tableau 9-2.

Tableau 9-2. Versions standard d'exclusion des robots

Version Titre et description Date

0.0 Une norme pour l'exclusion des robots — Mécanisme robots.txt original de Martijn Koster avec directive Disallow Juin 1994

1.0 Une méthode pour le contrôle des robots Web — Projet IETF de Martijn Koster avec prise en charge supplémentaire pour Allow, novembre 1996

2.0 Une norme étendue pour l'exclusion des robots — extension de Sean Conner incluant des informations sur les expressions régulières et le temps ; non largement prise en charge. Novembre 1996.

La plupart des robots actuels adoptent les normes v0.0 ou v1.0. La norme v2.0 est beaucoup plus complexe et n'a pas été largement adoptée. Elle pourrait ne jamais l'être. Nous nous concentrerons ici sur

la norme v1.0, car elle est largement utilisée et entièrement compatible avec la v0.0.

Sites Web et fichiers robots.txt

Avant de visiter une URL d'un site web, un robot doit récupérer et traiter le fichier robots.txt présent sur le site, s'il est présent.* Il existe une seule ressource robots.txt pour l'ensemble du site web, définie par le nom d'hôte et le numéro de port. Si le site est hébergé virtuellement, il peut y avoir un fichier robots.txt différent pour chaque docroot virtuel, comme pour tout autre fichier.

Actuellement, il n'est pas possible d'installer des fichiers robots.txt « locaux » dans des sous-répertoires individuels d'un site web. Il incombe au webmaster de créer un fichier robots.txt agrégé décrivant les règles d'exclusion applicables à l'ensemble du contenu du site web.

Récupération du fichier robots.txt

Les robots récupèrent la ressource robots.txt à l'aide de la méthode HTTP GET, comme tout autre fichier sur le serveur web. Le serveur renvoie le fichier robots.txt, s'il est présent, au format texte. Si le serveur renvoie un code HTTP 404 « Not Found », le robot peut supposer qu'il n'existe aucune restriction d'accès robotique et qu'il peut demander n'importe quel fichier.

Les robots doivent transmettre des informations d'identification dans les en-têtes « De » et « User-Agent » afin d'aider les administrateurs du site à suivre les accès robotisés et à fournir des coordonnées en cas de question ou de réclamation concernant le robot. Voici un exemple de requête HTTP d'un robot web commercial :

OBTENIR /robots.txt HTTP/1.0

Hébergeur : www.joes-hardware.com

Agent utilisateur : Slurp/2.0

Date : mer. 3 oct. 20:22:48 EST 2001

Codes de réponse

De nombreux sites web ne disposent pas de fichier robots.txt, mais le robot l'ignore. Il doit tenter de l'obtenir sur chaque site. Le robot exécute différentes actions selon le résultat de la récupération :

• Si le serveur répond avec un statut de réussite (code de statut HTTP 2XX), le robot doit analyser le contenu et appliquer les règles d'exclusion aux récupérations à partir de ce site.

• Si la réponse du serveur indique que la ressource n'existe pas (code d'état HTTP 404), le robot peut supposer qu'aucune règle d'exclusion n'est active et que l'accès au site n'est pas restreint par robots.txt.

* Même si nous parlons de « fichier robots.txt », il n'y a aucune raison pour que la ressource robots.txt réside strictement dans un système de fichiers. Par exemple, la ressource robots.txt pourrait être générée dynamiquement par une application passerelle.

• Si la réponse du serveur indique des restrictions d'accès (code d'état HTTP 401 ou 403), le robot doit considérer l'accès au site comme complètement restreint.

• Si la tentative de requête aboutit à un échec temporaire (code d'état HTTP 503), le robot doit reporter les visites sur le site jusqu'à ce que la ressource puisse être récupérée.

• Si la réponse du serveur indique une redirection (code d'état HTTP 3XX), le robot doit suivre les redirections jusqu'à ce que la ressource soit trouvée.

Format de fichier robots.txt

Le fichier robots.txt possède une syntaxe très simple, orientée lignes. Il comprend trois types de lignes : les lignes vides, les lignes de commentaires et les lignes de règles. Ces dernières ressemblent à des en-têtes HTTP (<Champ>: <valeur>) et sont utilisées pour la correspondance de motifs. Par exemple :

ce fichier robots.txt permet à Slurp & Webcrawler d'explorer # les parties publiques de notre site, mais aucun autre robot...

Agent utilisateur : slurp

Agent utilisateur : robot d'exploration Web

Interdire : /private

Agent utilisateur : * Interdire :

Les lignes d'un fichier robots.txt sont logiquement séparées en « enregistrements ». Chaque enregistrement décrit un ensemble de règles d'exclusion pour un groupe de robots spécifique. Ainsi, différentes règles d'exclusion peuvent être appliquées à différents robots.

Chaque enregistrement est constitué d'un ensemble de lignes de règles, terminées par une ligne vide ou un caractère de fin de fichier. Un enregistrement commence par une ou plusieurs lignes User-Agent, spécifiant les robots concernés, suivies de lignes Disallow et Allow indiquant les URL auxquelles ces robots peuvent accéder.*

L'exemple précédent montre un fichier robots.txt qui permet aux robots Slurp et Webcrawler d'accéder à tous les fichiers, à l'exception de ceux du sous-répertoire privé. Ce même fichier empêche également les autres robots d'accéder au contenu du site.

Examinons les lignes User-Agent, Disallow et Allow.

La ligne User-Agent

L'enregistrement de chaque robot commence par une ou plusieurs lignes User-Agent, de la forme :

Agent utilisateur : <nom-robot> ou :

Agent utilisateur : *

* Pour des raisons pratiques, le logiciel du robot doit être robuste et flexible avec le caractère de fin de ligne. CR, LF et CRLF doivent tous être pris en charge.

Le nom du robot (choisi par l'implémenteur du robot) est envoyé dans l'en-tête User-Agent de la requête HTTP GET du robot.

Lorsqu'un robot traite un fichier robots.txt, il doit respecter l'enregistrement avec :

• Le premier nom de robot qui est une sous-chaîne insensible à la casse du nom du robot

• Le premier nom de robot qui est « * »

Si le robot ne trouve pas de ligne User-Agent correspondant à son nom et ne trouve pas de ligne générique « User-Agent : * », aucun enregistrement ne correspond et l'accès est illimité.

Étant donné que le nom du robot correspond à des sous-chaînes insensibles à la casse, soyez vigilant aux fausses correspondances. Par exemple, « User-Agent: bot » correspond à tous les robots nommés Bot, Robot, Bottom-Feeder, Spambot et Dont-Bother-Me.

Les lignes Interdire et Autoriser

Les lignes « Disallow » et « Allow » suivent immédiatement les lignes « User-Agent » d'un enregistrement d'exclusion de robot. Elles décrivent les chemins d'URL explicitement interdits ou explicitement autorisés pour les robots spécifiés.

Le robot doit comparer l'URL souhaitée à toutes les règles d'autorisation et de refus de l'enregistrement d'exclusion, dans l'ordre. La première correspondance trouvée est utilisée. En l'absence de correspondance, l'URL est autorisée.*

Pour qu'une ligne Autoriser/Interdire corresponde à une URL, le chemin de la règle doit être un préfixe sensible à la casse du chemin d'URL. Par exemple, « Interdire : /tmp » correspond à toutes les URL suivantes :

http://www.joes-hardware.com/tmp http://www.joes-hardware.com/tmp/ http://www.joes-hardware.com/tmp/pliers.html http://www.joes-hardware.com/tmpspc/stuff.txt

Interdire/Autoriser la correspondance des préfixes

Voici quelques détails supplémentaires sur la correspondance des préfixes Interdire/Autoriser :

• Les règles d'interdiction et d'autorisation requièrent des correspondances de préfixes sensibles à la casse. L'astérisque n'a pas de signification particulière (contrairement aux lignes User-Agent), mais l'effet générique universel peut être obtenu à partir d'une chaîne vide.

• Tous les caractères « échappés » (%XX) dans le chemin de la règle ou le chemin de l'URL sont réintroduits en octets avant la comparaison (à l'exception de %2F, la barre oblique, qui doit correspondre exactement).

• Si le chemin de la règle est une chaîne vide, elle correspond à tout.

Le tableau 9-3 répertorie plusieurs exemples de correspondance entre les chemins de règles et les chemins d'URL.

* L'URL robots.txt est toujours autorisée et ne doit pas apparaître dans les règles Autoriser/Interdire.

Tableau 9-3. Exemples de correspondances de chemins de fichiers robots.txt

Chemin de règle Chemin URL Correspondance ? Commentaires

/tmp /tmp Chemin de la règle == Chemin de l'URL

/tmp /tmpfile.html Le chemin de la règle est un préfixe du chemin de l'URL

/tmp /tmp/a.html Le chemin de la règle est un préfixe du chemin de l'URL

/tmp/ /tmp ✗ /tmp/ n'est pas un préfixe de /tmp

README.TXT Le chemin de règle vide correspond à tout

/~fred/hi.html %7Efred/hi.html %7E est traité de la même manière que ~

/%7Efred/hi.html /~fred/hi.html %7E est traité de la même manière que ~

/%7efred/hi.html /%7Efred/hi.html La casse n'est pas significative dans les échappements

/~fred/hi.html ~fred%2Fhi.html ✗ %2F est une barre oblique, mais la barre oblique est un cas particulier qui doit correspondre exactement

La correspondance des préfixes fonctionne généralement bien, mais elle manque parfois d'efficacité. Si vous souhaitez également interdire l'exploration de certains sous-répertoires, quel que soit le préfixe du chemin, robots.txt ne propose aucune solution. Par exemple, vous

pourriez vouloir éviter l'exploration des sous-répertoires de contrôle de version RCS. La version 1.0 du schéma robots.txt ne permet pas cette opération, si ce n'est en énumérant séparément chaque chemin d'accès à chaque sous-répertoire RCS.

Autres robots.txt Sagesse

Voici quelques autres règles concernant l'analyse du fichier robots.txt :

• Le fichier robots.txt peut contenir d'autres champs que « User-Agent », « Disallow » et « Allow », à mesure que la spécification évolue. Un robot doit ignorer tout champ qu'il ne comprend pas.

• Pour des raisons de compatibilité descendante, la rupture de ligne n'est pas autorisée.

• Les commentaires sont autorisés n'importe où dans le fichier ; ils consistent en des espaces facultatifs, suivis d'un caractère de commentaire (#) suivi du commentaire, jusqu'au caractère de fin de ligne.

• La version 0.0 de la norme d'exclusion des robots ne prenait pas en charge la ligne « Allow ». Certains robots implémentent uniquement la spécification de la version 0.0 et ignorent les lignes « Allow ». Dans ce cas, un robot se comportera de manière conservatrice et ne récupérera pas les URL autorisées.

Mise en cache et expiration du fichier robots.txt

Si un robot devait récupérer un fichier robots.txt avant chaque accès, cela doublerait la charge des serveurs web et réduirait son efficacité. Les robots sont censés récupérer le fichier robots.txt régulièrement et mettre les résultats en cache. La copie en cache du fichier robots.txt doit être utilisée par le robot jusqu'à son expiration. Les mécanismes de contrôle de cache HTTP standard sont utilisés par le serveur d'origine et les robots pour contrôler la mise en cache du fichier robots.txt. Les robots doivent tenir compte des en-têtes Cache-Control et Expires dans la réponse HTTP.*

De nombreux robots d'exploration de production actuels ne sont pas des clients HTTP/1.1 ; les webmasters doivent noter que ces robots d'exploration ne comprendront pas nécessairement les directives de mise en cache fournies pour la ressource robots.txt.

En l'absence de directives Cache-Control, la spécification préliminaire autorise une mise en cache pendant sept jours. Mais, en pratique, cette durée est souvent trop longue. Les administrateurs de serveurs web qui ne connaissent pas le fichier robots.txt en créent souvent un en réponse à une visite robotique. Cependant, si l'absence de fichier robots.txt est mise en cache pendant une semaine, le nouveau fichier robots.txt apparaîtra comme sans effet, et l'administrateur du site

accusera l'administrateur robot de ne pas respecter la norme d'exclusion des robots.†.

Code Perl d'exclusion de robot

Il existe quelques bibliothèques Perl accessibles au public permettant d'interagir avec les fichiers robots.txt. Par exemple, le module WWW::RobotsRules, disponible pour l'archive Perl publique du CPAN.

Le fichier robots.txt analysé est conservé dans l'objet WWW::RobotRules, qui fournit des méthodes permettant de vérifier si l'accès à une URL donnée est interdit. Le même WWW::

L'objet RobotRules peut analyser plusieurs fichiers robots.txt.

Voici les principales méthodes de l'API WWW::RobotRules : Créer un objet RobotRules

$rules = WWW::RobotRules->new($robot_name);

Charger le fichier robots.txt

$rules->parse($url, $content, $fresh_until);

Vérifier si l'URL d'un site est récupérable

$can_fetch = $rules->allowed($url);

Voici un court programme Perl qui démontre l'utilisation de WWW::RobotRules :

nécessite WWW::RobotRules ;

Créez l'objet RobotRules en nommant le robot "SuperRobot" my $robotsrules = new WWW::RobotRules 'SuperRobot/1.0'; use LWP::Simple qw(get);

Obtenir et analyser le fichier robots.txt pour Joe's Hardware, en accumulant les règles

$url = "http://www.joes-hardware.com/robots.txt"; mon $robots_txt = obtenir $url; $robotsrules->parse($url, $robots_txt);

* Voir « Conserver les copies à jour » au chapitre 7 pour en savoir plus sur la gestion des directives de mise en cache.

† Plusieurs robots d'exploration Web à grande échelle utilisent la règle de récupération quotidienne du fichier robots.txt lorsqu'ils explorent activement le Web. # Récupérez et analysez le fichier robots.txt de Mary's Antiques, en accumulant les règles

$url = "http://www.marys-antiques.com/robots.txt"; mon $robots_txt = obtenir $url; $robotsrules->parse($url, $robots_txt);

Maintenant, RobotRules contient l'ensemble des règles d'exclusion de robots pour plusieurs

différents sites. Cela les sépare tous. Nous pouvons maintenant utiliser RobotRules.

pour tester si un robot est autorisé à accéder à différentes URL.

si ($robotsrules->allowed($some_target_url))

{

  $c = obtenir $url; ...

}

Ce qui suit est un fichier robots.txt hypothétique pour www.marys-antiques.com :

##
###########

Ceci est le fichier robots.txt du site Web de Mary's Antiques

##
###########

Gardez le robot de Suzy hors de toutes les URL dynamiques car il ne

les comprendre, et parmi toutes les données privées, à l'exception de la # petite section que Mary a réservée sur le site pour Suzy.

Agent utilisateur : Suzy-Spider

Interdire : /dynamic

Autoriser : /private/suzy-stuff

Interdire : /private

Le robot Furniture-Finder a été spécialement conçu pour comprendre

Programme d'inventaire de meubles du magasin d'antiquités de Mary, alors laissez-le

explorez cette ressource, mais gardez-la hors de toutes les autres ressources dynamiques et hors de toutes les données privées.

Agent utilisateur : Furniture-Finder

Autoriser : /dynamic/check-inventory

Interdire : /dynamic

Interdire : /private # Empêcher tout le monde d’accéder aux passerelles dynamiques et aux données privées.

Agent utilisateur : *

Interdire : /dynamic

Interdire : /private

Ce fichier robots.txt contient un enregistrement pour le robot SuzySpider, un autre pour le robot FurnitureFinder et un enregistrement par défaut pour tous les autres robots. Chaque enregistrement applique des politiques d'accès différentes aux différents robots :

• L'enregistrement d'exclusion pour SuzySpider empêche le robot d'explorer les URL de passerelle d'inventaire du magasin qui commencent par /dynamic et hors des données utilisateur privées, à l'exception de la section réservée à Suzy.

• L'enregistrement du robot FurnitureFinder lui permet d'explorer l'URL de la passerelle d'inventaire de meubles. Ce robot comprend peut-être le format et les règles de la passerelle de Mary.

• Tous les autres robots sont tenus à l'écart de toutes les pages Web dynamiques et privées, bien qu'ils puissent explorer le reste des URL.

Le tableau 9-4 répertorie quelques exemples de différentes accessibilités de robots au site Web de Mary's Antiques.

Tableau 9-4. Accessibilité du robot au site web de Mary's Antiques

URL SuzySpider FurnitureFinder NosyBot

http://www.marys-antiques.com/

http://www.marys-antiques.com/index.html

http://www.marys-antiques.com/private/payroll.xls ✗ ✗ ✗

http://www.marys-antiques.com/private/suzy-stuff/taxes.txt ✗ ✗

http://www.marys-antiques.com/dynamic/buy-stuff?id=3546 ✗ ✗ ✗

http://www.marys-antiques.com/dynamic/check-inventory?kitchen ✗ ✗

Balises META HTML Robot-Control

Le fichier robots.txt permet à l'administrateur d'un site d'exclure les robots de tout ou partie d'un site web. L'un des inconvénients de ce fichier est qu'il appartient à l'administrateur du site web, et non à l'auteur du contenu.

Les auteurs de pages HTML disposent d'un moyen plus direct de restreindre l'accès des robots à certaines pages. Ils peuvent ajouter directement des balises de contrôle des robots aux documents HTML. Les robots qui respectent ces balises pourront toujours récupérer les documents, mais si une balise d'exclusion des robots est présente, ils les ignoreront. Par exemple, un robot de moteur de recherche Internet n'inclurait pas le document dans son index de recherche. Comme pour la norme robots.txt, la participation est encouragée, mais non imposée.

Les balises d'exclusion des robots sont implémentées à l'aide de balises HTML META, sous la forme :

<META NAME="ROBOTS" CONTENT=liste-de-directives>

Directives META du robot

Il existe plusieurs types de directives META pour les robots, et de nouvelles directives sont susceptibles d'être ajoutées au fil du temps, à mesure que les moteurs de recherche et leurs robots développent leurs activités et leurs fonctionnalités. Les deux directives META les plus fréquemment utilisées sont :

NOINDEX

Indique à un robot de ne pas traiter le contenu de la page et d'ignorer le document

(c'est-à-dire ne pas inclure le contenu dans un index ou une base de données).

<META NAME="ROBOTS" CONTENT="NOINDEX">

NOFOLLOW

Indique à un robot de ne pas explorer les liens sortants de la page.

<META NAME="ROBOTS" CONTENT="NOFOLLOW">

Outre NOINDEX et NOFOLLOW, il existe les directives opposées INDEX et FOLLOW, la directive NOARCHIVE, ainsi que les directives ALL et NONE. Ces directives de balises META robot sont résumées comme suit :

INDICE

Indique à un robot qu'il peut indexer le contenu de la page.

SUIVRE

Indique à un robot qu'il peut explorer tous les liens sortants de la page.

NOARCHIVE

Indique à un robot qu'il ne doit pas mettre en cache une copie locale de la page.*

TOUS

Équivalent à INDEX, FOLLOW.

AUCUN

Équivalent à NOINDEX, NOFOLLOW.

Les balises META du robot, comme toutes les balises META HTML, doivent apparaître dans la section HEAD d'une page HTML :

```
<html>
<tête>
   <meta name="robots" content="noindex,nofollow">
   <titre>...</titre>
</head>
<corps> ...
</body>
</html>
```

Notez que le nom « robots » de la balise et le contenu ne sont pas sensibles à la casse.

Vous ne devez évidemment pas spécifier de directives contradictoires ou répétitives, telles que :

```
<meta name="robots"
content="INDEX,NOINDEX,NOFOLLOW,FOLLOW,FOLLOW">
```

dont le comportement est probablement indéfini et variera certainement d'une implémentation de robot à une autre.

Balises META des moteurs de recherche

Nous venons de parler des balises META robots, utilisées pour contrôler l'exploration et l'indexation des robots web. Toutes les balises META robots contiennent l'attribut name="robots".

* Cette balise META a été introduite par les responsables du moteur de recherche Google afin de permettre aux webmasters de désactiver la diffusion de leurs contenus en cache par Google. Elle peut également être utilisée avec META NAME="googlebot".

De nombreux autres types de balises META sont disponibles, notamment ceux présentés dans le tableau 9-5. Les balises META DESCRIPTION et KEYWORDS sont utiles aux robots des moteurs de recherche pour l'indexation du contenu.

Tableau 9-5. Directives supplémentaires pour les balises META

nom=	contenu=	Description
DESCRIPTION	<texte>	Permet à l'auteur de définir un court résumé textuel de la page web. De nombreux moteurs de recherche utilisent les balises META DESCRIPTION, permettant aux auteurs de spécifier de courts résumés appropriés pour décrire leurs pages web.

```
<meta name="description"
```

content="Bienvenue sur le site Web de Mary's Antiques">

MOTS CLÉS <liste de virgules> Associe une liste de mots séparés par des virgules qui décrivent la page Web, pour faciliter les recherches de mots clés.

<meta name="mots-clés"

contenu="antiquités,marie,meubles,restauration">

REVISIT-AFTER a <no. days> Indique au robot ou au moteur de recherche que la page doit être revisitée, probablement parce qu'elle est sujette à changement, après le nombre de jours spécifié.

<meta name="revisit-after" content="10 jours">

a Cette directive n’est pas susceptible de bénéficier d’un large soutien.

Étiquette des robots

En 1993, Martijn Koster, pionnier de la communauté des robots web, a rédigé une liste de conseils pour les auteurs de robots web. Si certains de ces conseils sont obsolètes, la plupart restent très utiles. Le traité original de Martijn, « Guidelines for Robot Writers », est disponible à l'adresse http://www.robotstxt.org/wc/guidelines.html.

Le tableau 9-6 propose une mise à jour moderne destinée aux concepteurs et opérateurs de robots, largement inspirée de l'esprit et du contenu de la liste originale. La plupart de ces directives sont destinées aux robots du World Wide Web ; cependant, elles s'appliquent également aux robots d'exploration de plus petite taille.

Tableau 9-6. Directives pour les opérateurs de robots Web

Description des lignes directrices

(1) Identification

Identifiez votre robot : utilisez le champ « User-Agent » HTTP pour indiquer aux serveurs web le nom de votre robot. Cela permettra aux administrateurs de comprendre ce que fait votre robot. Certains robots incluent également une URL décrivant leur fonction et leurs politiques dans l'en-tête « User-Agent ».

Identifiez votre machine : assurez-vous que votre robot fonctionne depuis une machine disposant d'une entrée DNS, afin que les sites web puissent inverser l'adresse IP du robot en nom d'hôte. Cela permettra à l'administrateur d'identifier l'organisation responsable du robot.

Identifier un contact Utilisez le champ HTTP De pour fournir une adresse e-mail de contact.

Étiquette des robots

Tableau 9-6. Directives pour les opérateurs de robots Web (suite)

Description des lignes directrices

(2) Opérations

Soyez vigilant. Votre robot suscitera des questions et des plaintes. Certaines sont causées par des robots égarés. Soyez vigilant et veillez au bon fonctionnement de votre robot. S'il fonctionne 24 heures sur 24, redoublez de prudence. Il se peut que des opérateurs le surveillent 24 heures sur 24, 7 jours sur 7, jusqu'à ce qu'il soit bien rodé.

Soyez prêt. Lorsque vous entreprenez un projet robotique majeur, veillez à en informer les membres de votre organisation. Votre organisation devra surveiller la consommation de bande passante du réseau et se tenir prête à répondre à toute demande de renseignements du public.

Surveiller et enregistrer votre robot doit être doté de nombreux outils de diagnostic et de journalisation, afin de suivre sa progression, d'identifier les pièges et de vérifier son bon fonctionnement. Nous ne saurions trop insister sur l'importance de surveiller et d'enregistrer le comportement d'un robot. Des problèmes et des plaintes surviendront, et disposer de journaux détaillés du comportement d'un robot peut aider l'opérateur à remonter la piste. Ceci est important non seulement pour déboguer votre robot d'exploration, mais aussi pour le protéger contre les plaintes injustifiées.

Apprendre et s'adapter À chaque exploration, vous apprendrez de nouvelles choses. Adaptez votre robot pour qu'il s'améliore à chaque fois et évite les pièges courants.

(3) Limitez-vous

Filtrer par URL Si une URL semble renvoyer à des données que vous ne comprenez pas ou qui ne vous intéressent pas, vous pouvez l'ignorer. Par exemple, les URL se terminant par « .Z », « .gz », « .tar » ou « .zip » sont susceptibles d'être des fichiers compressés ou des archives. Les URL se terminant par « .exe » sont susceptibles d'être des programmes. Les URL se terminant par « .gif », « .tif », « .jpg » sont susceptibles d'être des images. Assurez-vous d'obtenir ce que vous cherchez.

Filtrer les URL dynamiques. En général, les robots ne souhaitent pas explorer le contenu des passerelles dynamiques. Ils ne sauront pas comment procéder.

formater et envoyer correctement les requêtes aux passerelles, et les résultats risquent d'être irréguliers ou transitoires. Si une URL contient « cgi » ou un « ? », le robot préférera peut-être éviter de l'explorer.

Filtrer avec les en-têtes Accept Votre robot doit utiliser les en-têtes HTTP Accept pour indiquer aux serveurs quel type de contenu il comprend.

Adhérez au fichier robots.txt Votre robot doit adhérer aux contrôles du fichier robots.txt sur le site.

Limitez vos accès : Votre robot doit compter le nombre d'accès à chaque site et le moment où ils ont eu lieu, et utiliser ces informations pour s'assurer qu'il ne visite pas un site trop fréquemment. Lorsqu'un robot accède à un site plus fréquemment qu'une fois toutes les quelques minutes, les administrateurs deviennent méfiants. Lorsqu'un robot accède à un site toutes les quelques secondes, certains administrateurs s'énervent. Lorsqu'un robot martèle un site aussi vite qu'il le peut, bloquant tout autre trafic, les administrateurs seront furieux.

En général, limitez votre robot à quelques requêtes par minute maximum, espacées de quelques secondes. Limitez également le nombre total d'accès à un site afin d'éviter les boucles.

(4) Tolérer les boucles, les doublons et autres problèmes

Gérer tous les codes de retour : Vous devez être prêt à gérer tous les codes d'état HTTP, y compris les redirections et les erreurs. Vous devez également consigner et surveiller ces codes. Un grand nombre de résultats d'échec sur un site doit faire l'objet d'une enquête. Il se peut que de nombreuses URL soient obsolètes ou que le serveur refuse de fournir des documents aux robots.

Canonicaliser les URL Essayez de supprimer les alias courants en normalisant toutes les URL dans une forme standard.

Évitez les cycles de manière proactive. Détectez et évitez les cycles. Considérez le processus d'exploration comme une boucle de rétroaction. Les résultats des problèmes et leurs résolutions doivent être répercutés sur l'exploration suivante, améliorant ainsi votre robot à chaque itération.

Tableau 9-6. Directives pour les opérateurs de robots Web (suite)

Description des lignes directrices

Surveillez les pièges. Certains types de cycles sont intentionnels et malveillants. Ils peuvent être intentionnellement difficiles à détecter. Surveillez les nombreux accès à un site avec des URL étranges. Il peut s'agir de pièges.

Maintenir une liste noire Lorsque vous trouvez des pièges, des cycles, des sites cassés et des sites qui veulent que votre robot reste à l'écart, ajoutez-les à une liste noire et ne les visitez plus.

(5) Évolutivité Comprendre l'espace Calculez à l'avance l'ampleur du problème que vous résolvez. Vous pourriez être surpris de la quantité de mémoire dont votre application aura besoin pour réaliser une tâche robotique, en raison de l'ampleur du Web.

Comprendre la bande passante : déterminez la bande passante réseau dont vous disposez et celle dont vous aurez besoin pour réaliser votre tâche robotique dans les délais impartis. Surveillez l'utilisation réelle de la bande passante réseau. Vous constaterez probablement que la bande passante sortante (requêtes) est bien inférieure à la bande passante entrante (réponses). En surveillant l'utilisation du réseau, vous pouvez également identifier des pistes pour optimiser votre robot et lui permettre de mieux exploiter la bande passante réseau grâce à une meilleure utilisation de ses connexions TCP.

Comprendre le temps : déterminez le temps nécessaire à votre robot pour accomplir sa tâche et vérifiez que la progression correspond à votre estimation. Si votre robot est très loin de votre estimation, il y a probablement un problème qui mérite d'être examiné.

Diviser pour mieux régner Pour les explorations à grande échelle, vous devrez probablement utiliser davantage de matériel pour effectuer le travail, soit en utilisant de gros serveurs multiprocesseurs avec plusieurs cartes réseau, soit en utilisant plusieurs ordinateurs plus petits fonctionnant à l'unisson.

(6) Fiabilité

Testez minutieusement votre robot en interne avant de le déployer sur le monde. Lorsque vous êtes prêt à effectuer des tests hors site, effectuez d'abord quelques petits vols inauguraux. Recueillez de nombreux résultats et analysez vos performances et votre utilisation de la mémoire, en estimant leur évolutivité face à un problème plus vaste.

Point de contrôle : tout robot sérieux doit enregistrer un instantané de sa progression, à partir duquel il peut redémarrer en cas de panne. Des pannes sont inévitables : des bugs logiciels et des pannes matérielles. Les robots à grande échelle ne peuvent pas repartir de zéro à chaque fois. Prévoyez dès le départ une fonctionnalité de point de contrôle/redémarrage.

Résilience aux pannes Anticipez les pannes et concevez votre robot pour qu'il puisse continuer à progresser lorsqu'elles surviennent.

(7) Relations publiques

Soyez prêt. Votre robot risque de contrarier certaines personnes. Soyez prêt à répondre rapidement à leurs questions. Rédigez une déclaration de politique de confidentialité décrivant votre robot et incluez des instructions détaillées sur la création d'un fichier robots.txt.

Soyez compréhensif. Certaines personnes qui vous contactent au sujet de votre robot seront bien informées et encourageantes ; d'autres seront naïves. Quelques-unes seront particulièrement en colère. Certaines peuvent même paraître folles. Il est généralement improductif de discuter de l'importance de votre projet robotique. Expliquez-leur la norme d'exclusion des robots et, s'ils sont toujours

mécontents, supprimez immédiatement les URL plaignantes de votre exploration et ajoutez-les à la liste noire.

Soyez réactif. La plupart des webmasters mécontents manquent simplement de clarté sur les robots. Si vous répondez immédiatement et professionnellement, 90 % des plaintes disparaîtront rapidement. En revanche, si vous attendez plusieurs jours avant de répondre, alors que votre robot continue de visiter un site, attendez-vous à rencontrer un adversaire très virulent et en colère.

a Voir le chapitre 4 pour plus d'informations sur l'optimisation des performances TCP.

Étiquette des robots

Moteurs de recherche

Les robots web les plus répandus sont ceux utilisés par les moteurs de recherche. Ces derniers permettent aux utilisateurs de trouver des documents sur n'importe quel sujet, partout dans le monde.

Nombre des sites les plus populaires du Web actuel sont des moteurs de recherche. Ils servent de point de départ à de nombreux internautes et leur offrent un service précieux : ils les aident à trouver l'information qui les intéresse.

Les robots d'indexation alimentent les moteurs de recherche en récupérant les documents présents sur le Web et en leur permettant de créer des index répertoriant les mots clés présents dans les documents, à l'instar de l'index figurant à la fin de ce livre. Les moteurs de recherche sont la principale source de robots d'indexation web ; examinons rapidement leur fonctionnement.

Pensez grand

À ses débuts, les moteurs de recherche étaient des bases de données relativement simples qui aidaient les utilisateurs à localiser des documents sur le Web. Aujourd'hui, avec les milliards de pages accessibles sur le Web, les moteurs de recherche sont devenus essentiels pour aider les internautes à trouver des informations. Ils sont également devenus très complexes, car ils ont dû évoluer pour gérer l'ampleur du Web.

Avec des milliards de pages Web et des millions d'utilisateurs à la recherche d'informations, les moteurs de recherche doivent déployer des robots d'exploration sophistiqués pour récupérer ces milliards de pages Web, ainsi que des moteurs de requête sophistiqués pour gérer la charge de requêtes générée par des millions d'utilisateurs.

Imaginez la tâche d'un robot d'indexation web de production, qui doit exécuter des milliards de requêtes HTTP pour récupérer les pages nécessaires à l'index de recherche. Si chaque requête prenait une demi-seconde (ce qui est probablement lent pour certains serveurs et

rapide pour d'autres*), cela prend quand même (pour un milliard de documents) :

0,5 seconde × (1 000 000 000) / ((60 s/jour) × (60 min/heure) × (24 h/jour)), soit environ 5 700 jours si les requêtes sont effectuées séquentiellement ! De toute évidence, les robots d'exploration à grande échelle doivent être plus intelligents, paralléliser les requêtes et utiliser plusieurs machines pour effectuer la tâche. Cependant, en raison de son ampleur, explorer l'intégralité du Web reste un défi de taille.

Architecture moderne des moteurs de recherche

Les moteurs de recherche actuels créent des bases de données locales complexes, appelées « index de texte intégral », sur les pages web du monde entier et leur contenu. Ces index fonctionnent comme une sorte de catalogue sur fiches de tous les documents du Web.

* Cela dépend des ressources du serveur, du robot client et du réseau entre les deux.

Les robots d'exploration des moteurs de recherche collectent les pages web et les intègrent à l'index de texte intégral. Parallèlement, les utilisateurs des moteurs de recherche interrogent l'index de texte intégral via des passerelles de recherche telles que HotBot (http://www.hotbot.com) ou Google (http://www.google.com). Étant donné l'évolution constante des pages web et le temps nécessaire à l'exploration d'une grande partie du Web, l'index de texte intégral est au mieux une photographie du Web.

L’architecture de haut niveau d’un moteur de recherche moderne est illustrée dans la figure 9-7.

Figure 9-7. Un moteur de recherche de production contient des robots d'exploration et des passerelles de requêtes coopérants.

Index du texte intégral

Un index de texte intégral est une base de données qui identifie instantanément tous les documents contenant un mot. Il n'est pas nécessaire d'analyser les documents après la création de l'index.

La figure 9-8 présente trois documents et l'index de texte intégral correspondant. Cet index répertorie les documents contenant chaque mot.

Par exemple:

• Le mot « a » se trouve dans les documents A et B.

• Le mot « meilleur » se trouve dans les documents A et C.

• Le mot « drill » se trouve dans les documents A et B.

• Le mot « routine » se trouve dans les documents B et C.

• Le mot « le » est présent dans les trois documents, A, B et C.

Moteurs de recherche

Figure 9-8. Trois documents et un index intégral

Publication de la requête

Lorsqu'un utilisateur adresse une requête à une passerelle de moteur de recherche web, il remplit un formulaire HTML et son navigateur l'envoie à la passerelle via une requête HTTP GET ou POST. La passerelle extrait la requête et convertit l'interface web.

interroger l'expression utilisée pour rechercher l'index de texte intégral.*

La figure 9-9 illustre une requête utilisateur simple sur le site www.joes-hardware.com. L'utilisateur saisit « drills » dans le champ de recherche, et le navigateur traduit cette requête en une requête GET dont le paramètre de requête est intégré à l'URL.† Le serveur web de Joe's Hardware reçoit la requête et la transmet à sa passerelle de recherche, qui renvoie la liste de documents obtenue au serveur web, qui la formate ensuite dans une page HTML pour l'utilisateur.

Trier et présenter les résultats

Une fois qu'un moteur de recherche a utilisé son index pour déterminer les résultats d'une requête, l'application passerelle prend les résultats et crée une page de résultats pour l'utilisateur final.

* La méthode de transmission de cette requête dépend de la solution de recherche utilisée.

† « Chaînes de requête » au chapitre 2 décrit l’utilisation courante du paramètre de requête dans les URL.

Figure 9-9. Exemple de requête de recherche

Comme de nombreuses pages web peuvent contenir n'importe quel mot, les moteurs de recherche déploient des algorithmes intelligents pour classer les résultats. Par exemple, dans la figure 9-8, le mot « meilleur » apparaît dans plusieurs documents ; les moteurs de recherche doivent connaître l'ordre dans lequel ils doivent présenter la liste des documents de résultats afin de proposer aux utilisateurs les résultats les plus pertinents. C'est ce qu'on appelle le classement par pertinence : le processus de notation et d'ordonnancement d'une liste de résultats de recherche.

Pour faciliter ce processus, de nombreux grands moteurs de recherche utilisent des données de recensement collectées lors de l'exploration du Web. Par exemple, compter le nombre de liens pointant vers une page donnée peut aider à déterminer sa popularité, et cette information peut être utilisée pour pondérer l'ordre dans lequel les résultats sont présentés. Les algorithmes, les astuces d'exploration et autres astuces utilisées par les moteurs de recherche comptent parmi leurs secrets les mieux gardés.

Usurpation d'identité

Les utilisateurs étant souvent frustrés de ne pas trouver ce qu'ils recherchent dès les premiers résultats d'une recherche, l'ordre des résultats peut être important pour trouver un site. Les webmasters ont tout intérêt à placer leurs sites en haut des résultats pour les mots-clés qui, selon eux, les décrivent le mieux.

Moteurs de recherche

leurs sites, en particulier s’ils sont commerciaux et comptent sur les utilisateurs pour les trouver et utiliser leurs services.

Cette volonté d'améliorer le référencement a conduit à de nombreuses manipulations du système de recherche et a créé un bras de fer constant entre les développeurs de moteurs de recherche et ceux qui cherchent à faire figurer leurs sites en bonne place. De nombreux webmasters listent des tonnes de mots-clés (dont certains sont hors de propos) et déploient de fausses pages, ou des spoofs, voire des applications passerelles générant de fausses pages susceptibles de tromper les algorithmes de pertinence des moteurs de recherche pour certains mots.

En conséquence, les développeurs de moteurs de recherche et de robots doivent constamment ajuster leurs algorithmes de pertinence pour mieux détecter ces usurpations.

Pour plus d'informations

Pour plus d'informations sur les clients Web, reportez-vous à :

http://www.robotstxt.org/wc/robots.html

Les pages Web Robots : ressources pour les développeurs de robots, y compris le registre des robots Internet.

http://www.searchengineworld.com

Search Engine World : ressources pour les moteurs de recherche et les robots.

http://www.searchtools.com

Outils de recherche pour sites Web et intranets : ressources pour les outils de recherche et les robots.

http://search.cpan.org/doc/ILYAZ/perl_ste/WWW/RobotRules.pm
Source Perl de RobotRules.

http://www.conman.org/people/spc/robots2.html

Une norme étendue pour l'exclusion des robots.

Gestion des gigaoctets : compression et indexation de documents et d'images Witten, I., Moffat, A., et Bell, T., Morgan Kaufmann.

Chapitre 10Ceci est le titre du livre CHAPITRE 10

HTTP-NG

Alors que ce livre touche à sa fin, HTTP fête son dixième anniversaire. Et ce fut une décennie remarquable pour ce protocole Internet. Aujourd'hui, HTTP achemine la majeure partie du trafic numérique mondial.

Mais à mesure que HTTP entre dans sa phase d'adolescence, il est confronté à quelques défis. D'une certaine manière, son adoption a dépassé sa conception. Aujourd'hui, HTTP sert de base à de nombreuses applications diverses, sur de nombreuses technologies réseau différentes.

Ce chapitre présente certaines tendances et défis pour l'avenir de HTTP, ainsi qu'une proposition d'architecture de nouvelle génération appelée HTTP-NG. Bien que le groupe de travail sur HTTP-NG ait été dissous et que son adoption rapide semble désormais improbable, il esquisse néanmoins quelques orientations potentielles pour HTTP.

Les difficultés de croissance du protocole HTTP

À l'origine, HTTP était conçu comme une technique simple permettant d'accéder à du contenu multimédia lié à partir de serveurs d'information distribués. Mais, au cours de la dernière décennie, HTTP et ses dérivés ont acquis un rôle beaucoup plus large.

HTTP/1.1 offre désormais le balisage et la prise d'empreintes digitales pour le suivi des versions de documents, des méthodes pour le téléchargement de documents et les interactions avec les passerelles programmatiques, la prise en charge du contenu multilingue, la sécurité et l'authentification, la mise en cache pour réduire le trafic, le pipelining pour réduire la latence, les connexions persistantes pour réduire le temps de démarrage et améliorer la bande passante, et les accès par plage pour implémenter des mises à jour partielles. Les extensions et dérivés de HTTP sont allés encore plus loin, prenant en charge la publication de documents, le service d'applications, la messagerie arbitraire, le streaming vidéo et les bases de l'accès

multimédia sans fil. HTTP devient une sorte de « système d'exploitation » pour les applications multimédias distribuées.

La conception de HTTP/1.1, bien que bien pensée, commence à montrer des difficultés à mesure que HTTP est de plus en plus utilisé comme support unifié pour des opérations distantes complexes. HTTP présente au moins quatre points faibles :

Complexité

HTTP est un protocole relativement complexe et ses fonctionnalités sont interdépendantes. Implémenter correctement un logiciel HTTP est particulièrement complexe et source d'erreurs, en raison de la complexité et de l'imbrication des exigences et de l'imbrication de la gestion des connexions, de la gestion des messages et de la logique fonctionnelle.

Extensibilité

HTTP est difficile à étendre progressivement. De nombreuses applications HTTP héritées créent des incompatibilités avec les extensions de protocole, car elles ne disposent pas de technologie permettant des extensions de fonctionnalités autonomes.

Performance

Le protocole HTTP présente des problèmes de performances. Nombre de ces problèmes s'aggraveront avec l'adoption généralisée de technologies d'accès sans fil à forte latence et faible débit.

Dépendance aux transports

HTTP est conçu autour d'une pile réseau TCP/IP. Bien qu'il n'existe aucune restriction concernant les sous-piles alternatives, peu de travaux ont été menés dans ce domaine. HTTP doit mieux prendre en charge ces sous-piles alternatives pour devenir une plateforme de messagerie plus complète dans les applications embarquées et sans fil.

Activité HTTP-NG

À l'été 1997, le World Wide Web Consortium a lancé un projet spécial visant à étudier et à proposer une nouvelle version majeure de HTTP, capable de résoudre les problèmes de complexité, d'extensibilité, de performances et de dépendance au transport. Cette nouvelle version de HTTP a été baptisée HTTP : The Next Generation (HTTP-NG).

Un ensemble de propositions HTTP-NG a été présenté lors d'une réunion de l'IETF en décembre 1998. Ces propositions esquissaient une évolution majeure possible de HTTP. Cette technologie n'a pas été largement déployée (et ne le sera peut-être jamais), mais HTTP-NG représente l'effort le plus sérieux pour étendre la lignée de HTTP. Examinons HTTPNG plus en détail.

Modulariser et améliorer

Le thème de HTTP-NG peut être résumé en trois mots : « modulariser et améliorer ». Au lieu de mélanger la gestion des connexions, la gestion des messages, la logique de traitement du serveur et les méthodes de protocole, le groupe de travail HTTP-NG a proposé de modulariser le protocole en trois couches, illustrées à la figure 10-1 :

• La couche 1, la couche de transport des messages, se concentre sur la distribution de messages opaques entre les points d'extrémité, quelle que soit leur fonction. Elle prend en charge diverses sous-piles (par exemple, des piles pour les environnements sans fil) et se concentre sur les problèmes de distribution et de traitement efficaces des messages. L'équipe du projet HTTP-NG a proposé un protocole appelé WebMUX pour cette couche.

• La couche 2, couche d'invocation à distance, définit les fonctionnalités de requête/réponse permettant aux clients d'invoquer des opérations sur les ressources du serveur. Cette couche est indépendante du transport des messages et de la sémantique précise des opérations. Elle fournit simplement un moyen standard d'invoquer toute opération serveur. Cette couche vise à fournir un cadre extensible et orienté objet, plus proche de CORBA, DCOM et Java RMI que des méthodes statiques définies par le serveur de HTTP/1.1. L'équipe du projet HTTP-NG a proposé le protocole Binary Wire pour cette couche.

• La couche 3, la couche application web, fournit la majeure partie de la logique de gestion de contenu. Toutes les méthodes HTTP/1.1 (GET, POST, PUT, etc.), ainsi que les paramètres d'en-tête HTTP/1.1, y sont définis. Cette couche prend également en charge d'autres services basés sur l'invocation à distance, tels que WebDAV.

Figure 10-1. HTTP-NG sépare les fonctions en couches

Une fois les composants HTTP modularisés, ils peuvent être améliorés pour offrir de meilleures performances et des fonctionnalités plus riches.

Objets distribués

Une grande partie de la philosophie et des objectifs fonctionnels de HTTP-NG s'inspire largement des systèmes d'objets distribués structurés, orientés objet, tels que CORBA et DCOM. Ces systèmes peuvent contribuer à l'extensibilité et à la fonctionnalité des fonctionnalités.

Depuis 1996, une communauté de chercheurs plaide en faveur d'une convergence entre HTTP et des systèmes d'objets distribués plus sophistiqués. Pour en savoir plus sur les mérites d'un paradigme d'objets distribués pour le Web, consultez l'article de Xerox PARC intitulé « Migrating the Web Toward Distributed Objects ».

(ftp://ftp.parc.xerox.com/pub/ilu/misc/webilu.html).

Objets distribués

La philosophie ambitieuse d'unification du Web et des objets distribués a suscité une certaine résistance à l'adoption de HTTP-NG dans certaines communautés. Certains anciens systèmes d'objets distribués souffraient d'une implémentation lourde et d'une complexité formelle. L'équipe du projet HTTP-NG a tenté de répondre à certaines de ces préoccupations dans les exigences.

Couche 1 : Messagerie

Examinons de plus près les trois couches de HTTP-NG, en commençant par la couche la plus basse. La couche de transport des messages assure la distribution efficace des messages, indépendamment de leur signification et de leur finalité. Elle fournit une API pour la messagerie, quelle que soit la pile réseau sous-jacente.

Cette couche se concentre sur l'amélioration des performances de la messagerie, notamment :

• Pipelining et traitement par lots des messages pour réduire la latence aller-retour

• Réutiliser les connexions pour réduire la latence et améliorer la bande passante délivrée

• Multiplexage de plusieurs flux de messages en parallèle, sur la même connexion, pour optimiser les connexions partagées tout en évitant la pénurie de flux de messages

• Segmentation efficace des messages pour faciliter la détermination des limites des messages

L'équipe HTTP-NG a investi une grande partie de son énergie dans le développement du protocole WebMUX pour le transport de messages de couche 1. WebMUX est un protocole de messagerie haute performance qui fragmente et entrelace les messages sur une connexion TCP multiplexée. Nous aborderons WebMUX plus en détail plus loin dans ce chapitre.

Couche 2 : invocation à distance

La couche intermédiaire de l'architecture HTTP-NG prend en charge l'invocation de méthodes à distance. Cette couche fournit un cadre générique de requêtes/réponses permettant aux clients d'invoquer des opérations sur les ressources du serveur. Cette couche ne s'occupe pas de l'implémentation ni de la sémantique des opérations spécifiques (mise en cache, sécurité, logique des méthodes, etc.) ; elle se concentre uniquement sur l'interface permettant aux clients d'invoquer à distance des opérations du serveur.

De nombreuses normes d'invocation de méthodes distantes sont déjà disponibles (CORBA, DCOM et Java RMI, pour n'en citer que quelques-unes), et cette couche n'est pas destinée à prendre en charge toutes les fonctionnalités intéressantes de ces systèmes. Cependant, l'objectif explicite est d'étendre la prise en charge HTTP RMI par rapport à celle offerte par HTTP/1.1. Plus précisément, l'objectif est de fournir une prise en charge plus générale des appels de procédures distantes, de manière extensible et orientée objet.

L'équipe HTTP-NG a proposé le protocole Binary Wire pour cette couche. Ce protocole prend en charge une technologie extensible et performante permettant d'invoquer des opérations bien décrites sur un serveur et de récupérer les résultats. Nous aborderons le protocole Binary Wire plus en détail plus loin dans ce chapitre.

Couche 3 : application Web

La couche applicative web est le lieu où s'exécutent la sémantique et la logique spécifique à l'application. Le groupe de travail HTTP-NG a résisté à la tentation d'étendre les fonctionnalités des applications HTTP, se concentrant plutôt sur l'infrastructure formelle.

La couche applicative web décrit un système fournissant des services spécifiques à chaque application. Ces services ne sont pas monolithiques ; différentes API peuvent être disponibles pour différentes applications. Par exemple, l'application web pour HTTP/1.1 serait différente de WebDAV, même si elles peuvent partager certains éléments. L'architecture HTTP-NG permet à plusieurs applications de coexister à ce niveau, partageant des fonctionnalités sous-jacentes, et fournit un mécanisme d'ajout de nouvelles applications.

La philosophie de la couche applicative web est de fournir des fonctionnalités équivalentes à celles des interfaces HTTP/1.1 et des extensions, tout en les remodelant en un cadre d'objets distribués extensibles. Pour en savoir plus sur les interfaces de la couche applicative web, consultez le site http://www.w3.org/Protocols/HTTP-NG/1998/08/draft-larner-nginterfaces-00.txt.

WebMUX

Le groupe de travail HTTP-NG a consacré une grande partie de son énergie au développement de la norme WebMUX pour le transport de messages. WebMUX est un système de messagerie sophistiqué et performant, permettant le transport parallèle de messages via une connexion TCP multiplexée. Les flux de messages individuels, produits et consommés à des débits différents, peuvent être efficacement mis en paquets et multiplexés via une seule ou un petit nombre de connexions TCP (voir Figure 10-2).

Figure 10-2. WebMUX peut multiplexer plusieurs messages sur une seule connexion.

Voici quelques-uns des objectifs importants du protocole WebMUX :

• Conception simple.

• Haute performance.

• Multiplexage : plusieurs flux de données (de protocoles arbitraires de niveau supérieur) peuvent être entrelacés de manière dynamique et efficace sur une seule connexion, sans bloquer les données en attendant les producteurs lents.

WebMUX

Contrôle de flux basé sur le crédit : les données sont produites et consommées à des rythmes différents, et les expéditeurs et les destinataires disposent de quantités de mémoire et de ressources CPU différentes. WebMUX utilise un système de contrôle de flux basé sur le crédit, où les destinataires annoncent à l'avance leur intérêt pour la réception des données afin d'éviter les blocages liés à la pénurie de ressources.

• Préservation de l'alignement : l'alignement des données est préservé dans le flux multiplexé afin que les données binaires puissent être envoyées et traitées efficacement.

• Fonctionnalités riches : l’interface est suffisamment riche pour prendre en charge une API de sockets.

Vous pouvez en savoir plus sur le protocole WebMUX sur http://www.w3.org/Protocols/MUX/WD-mux-980722.html.

Protocole de fil binaire

L'équipe HTTP-NG a proposé le protocole Binary Wire pour améliorer la manière dont le protocole HTTP de nouvelle génération prend en charge les opérations à distance.

HTTP-NG définit des « types d'objets » et attribue à chaque type d'objet une liste de méthodes. Chaque type d'objet se voit attribuer un URI, ce qui permet de publier sa description et ses méthodes. HTTP-NG propose ainsi un modèle d'exécution plus extensible et orienté objet que celui de HTTP/1.1, où toutes les méthodes étaient définies statiquement sur les serveurs.

Le protocole Binary Wire transmet les requêtes d'appel d'opération du client au serveur et les réponses de résultat d'opération du serveur au client via une connexion avec état. Cette connexion avec état offre une efficacité accrue.

Les messages de requête contiennent l'opération, l'objet cible et des valeurs de données facultatives. Les messages de réponse contiennent

l'état de fin de l'opération, le numéro de série de la requête correspondante (permettant un ordre arbitraire des requêtes et réponses parallèles) et des valeurs de retour facultatives. Outre les messages de requête et de réponse, ce protocole définit plusieurs messages de contrôle interne permettant d'améliorer l'efficacité et la robustesse de la connexion.

Vous pouvez en savoir plus sur le protocole Binary Wire sur http://www.w3.org/Protocols/HTTP-NG/1998/08/draft-janssen-httpng-wire-00.txt.

Statut actuel

Fin 1998, l'équipe HTTP-NG a conclu qu'il était trop tôt pour soumettre les propositions HTTP-NG à l'IETF en vue de leur normalisation. On craignait que l'industrie et la communauté ne se soient pas encore pleinement adaptées à HTTP/1.1 et que l'importante réarchitecture de HTTP-NG vers un paradigme d'objets distribués aurait été extrêmement perturbatrice sans un plan de transition clair.

Deux propositions ont été faites :

• Au lieu de tenter de promouvoir l'ensemble de la réarchitecture HTTP-NG en une seule étape, il a été proposé de se concentrer sur la technologie de transport WebMUX. Cependant, au moment de la rédaction de ce document, l'intérêt n'était pas suffisant pour créer un groupe de travail WebMUX.

• Des travaux ont été lancés pour déterminer si les types de protocoles formels peuvent être rendus suffisamment flexibles pour être utilisés sur le Web, peut-être grâce à XML. Ceci est particulièrement important pour un système d'objets distribués extensible. Ces travaux sont toujours en cours.

Au moment de la rédaction de ce document, aucun projet majeur de développement de HTTP-NG n'est en cours. Cependant, avec l'utilisation croissante de HTTP, son utilisation croissante comme plateforme pour diverses applications et l'adoption croissante des technologies Internet sans fil et grand public, certaines des techniques proposées dans le cadre du projet HTTP-NG pourraient s'avérer importantes pour l'adolescence de HTTP.

Pour plus d'informations

Pour plus d'informations sur HTTP-NG, veuillez vous référer aux spécifications détaillées et aux rapports d'activité suivants :

http://www.w3.org/Protocols/HTTP-NG/

Groupe de travail HTTP-NG (proposé), site Web du consortium W3C.

http://www.w3.org/Protocols/MUX/WD-mux-980722.html

« Le protocole WebMUX », par J. Gettys et H. Nielsen.

http://www.w3.org/Protocols/HTTP-NG/1998/08/draft-janssen-httpng-wire-00.txt « Protocole de câblage binaire pour HTTP-NG », par B. Janssen.

http://www.w3.org/Protocols/HTTP-NG/1998/08/draft-larner-nginterfaces-00.txt « Interfaces Web HTTP-NG », par D. Larner.

ftp://ftp.parc.xerox.com/pub/ilu/misc/webilu.html

« Migration du Web vers les objets distribués », par D. Larner.

Pour plus d'informations

PARTIE III

III. Identification, autorisation et sécurité

Les quatre chapitres de la troisième partie présentent une série de techniques et de technologies permettant de suivre l'identité, de renforcer la sécurité et de contrôler l'accès au contenu :

• Le chapitre 11, Identification du client et cookies, parle des techniques permettant d'identifier les utilisateurs, afin que le contenu puisse être personnalisé en fonction du public d'utilisateurs.

• Le chapitre 12, Authentification de base, met en évidence les mécanismes de base permettant de vérifier l'identité des utilisateurs. Ce chapitre examine également l'interface entre l'authentification HTTP et les bases de données.

• Le chapitre 13, Authentification Digest, explique l'authentification Digest, une amélioration complexe proposée à HTTP qui offre une sécurité considérablement améliorée.

• Le chapitre 14, HTTP sécurisé, est un aperçu détaillé de la cryptographie Internet, des certificats numériques et du Secure Sockets Layer (SSL).

Chapitre 11 CHAPITRE 11

Identification du client et cookies

Les serveurs web peuvent communiquer simultanément avec des milliers de clients différents. Ces serveurs ont souvent besoin de savoir à qui ils s'adressent, plutôt que de traiter toutes les requêtes comme

provenant de clients anonymes. Ce chapitre présente certaines technologies que les serveurs peuvent utiliser pour identifier leurs interlocuteurs.

La touche personnelle

HTTP était à l'origine un protocole de requête/réponse anonyme et sans état. Une requête provenait d'un client, était traitée par le serveur, et une réponse lui était renvoyée. Le serveur web disposait de peu d'informations pour déterminer l'auteur de la requête ou pour suivre la séquence de requêtes de l'utilisateur visiteur.

Les sites web modernes souhaitent apporter une touche personnelle. Ils souhaitent en savoir plus sur les utilisateurs à l'autre bout du fil et pouvoir suivre leur navigation. Les sites de vente en ligne populaires comme Amazon.com personnalisent leurs sites de plusieurs façons :

Salutations personnelles

Les messages de bienvenue et le contenu des pages sont générés spécialement pour l'utilisateur, afin de rendre l'expérience d'achat plus personnelle.

Recommandations ciblées

En connaissant les centres d'intérêt des clients, les magasins peuvent leur suggérer des produits qu'ils estiment susceptibles d'intéresser. Ils peuvent également proposer des offres spéciales anniversaire à l'occasion de leurs anniversaires et autres événements importants.

Informations administratives au dossier

Les acheteurs en ligne détestent devoir remplir sans cesse des formulaires fastidieux pour leur adresse et leur carte de crédit. Certains sites stockent ces informations administratives dans une base de données. Une fois identifiés, ils peuvent utiliser ces informations, ce qui simplifie considérablement l'expérience d'achat.

Suivi des sessions

Les transactions HTTP sont sans état. Chaque requête/réponse se produit de manière isolée. De nombreux sites web souhaitent créer un état incrémental à mesure que vous interagissez avec le site (par exemple, lorsque vous remplissez un panier d'achat). Pour ce faire, les sites web doivent pouvoir distinguer les transactions HTTP des différents utilisateurs.

Ce chapitre résume quelques-unes des techniques utilisées pour identifier les utilisateurs en HTTP. HTTP lui-même n'était pas doté d'un ensemble complet de fonctionnalités d'identification. Les premiers concepteurs de sites web (pratiques comme ils étaient) ont développé leurs propres technologies pour identifier les utilisateurs. Chaque

technique a ses forces et ses faiblesses. Dans ce chapitre, nous aborderons les mécanismes suivants pour identifier les utilisateurs :

• En-têtes HTTP qui contiennent des informations sur l'identité de l'utilisateur

• Suivi de l'adresse IP du client, pour identifier les utilisateurs par leur adresse IP

• Connexion utilisateur, en utilisant l'authentification pour identifier les utilisateurs

• URL Fat, une technique permettant d'intégrer l'identité dans les URL

• Les cookies, une technique puissante mais efficace pour maintenir une identité persistante

En-têtes HTTP

Le tableau 11-1 présente les sept en-têtes de requête HTTP qui contiennent le plus souvent des informations sur l'utilisateur. Nous allons maintenant aborder les trois premiers ; les quatre derniers sont utilisés pour des techniques d'identification plus avancées, que nous aborderons ultérieurement.

Tableau 11-1. Les en-têtes HTTP contiennent des informations sur les utilisateurs

Nom de l'en-tête Type d'en-tête Description

De la demande Adresse e-mail de l'utilisateur

Demande d'agent utilisateur Logiciel de navigation de l'utilisateur

Page de demande de référencement provenant de l'utilisateur en suivant le lien

Demande d'autorisation Nom d'utilisateur et mot de passe (discutés plus tard)

Extension IP du client (requête) Adresse IP du client (discutée plus tard)

Adresse IP du client (requête) de l'extension X-Forwarded-For (discutée plus tard)

Extension de cookie (requête) Étiquette d'identification générée par le serveur (discutée plus tard)

L'en-tête « De » contient l'adresse e-mail de l'utilisateur. Idéalement, il s'agirait d'une source d'identification fiable, car chaque utilisateur aurait une adresse e-mail différente. Cependant, peu de navigateurs envoient des en-têtes « De », par crainte que des serveurs peu scrupuleux collectent les adresses e-mail et les utilisent pour la distribution de courriers indésirables. En pratique, les en-têtes « De » sont envoyés par des robots d'indexation automatisés, de sorte qu'en

cas de problème, un webmaster dispose d'un moyen de signaler ses éventuels problèmes par e-mail.

L'en-tête User-Agent fournit au serveur des informations sur le navigateur utilisé par l'utilisateur, notamment le nom et la version du programme, et souvent des informations sur le système d'exploitation. Cela est parfois utile pour personnaliser le contenu afin qu'il interagisse correctement avec certains navigateurs et leurs attributs, mais cela ne permet pas d'identifier clairement l'utilisateur. Voici deux en-têtes User-Agent, l'un envoyé par Netscape Navigator et l'autre par Microsoft Internet Explorer :

Navigateur 6.2

Agent utilisateur : Mozilla/5.0 (Windows ; U ; Windows NT 5.0 ; en-US ; rv : 0.9.4) Gecko/20011128

Netscape6/6.2.1

Internet Explorer 6.01

Agent utilisateur : Mozilla/4.0 (compatible ; MSIE 6.0 ; Windows NT 5.0)

L'en-tête Referer fournit l'URL de la page d'origine de l'utilisateur. Il ne permet pas à lui seul d'identifier directement l'utilisateur, mais indique la page précédemment consultée. Vous pouvez l'utiliser pour mieux comprendre le comportement de navigation et les centres d'intérêt des utilisateurs. Par exemple, si vous arrivez sur un serveur web depuis un site de baseball, le serveur peut en déduire que vous êtes un supporter.

Les en-têtes « From », « User-Agent » et « Referer » ne permettent pas une identification fiable. Les sections suivantes présentent des schémas plus précis pour identifier des utilisateurs spécifiques.

Adresse IP du client

Les premiers pionniers du web ont tenté d'utiliser l'adresse IP du client comme moyen d'identification. Ce système fonctionne si chaque utilisateur possède une adresse IP distincte, si cette adresse change rarement (voire jamais) et si le serveur web peut déterminer l'adresse IP du client pour chaque requête. Bien que l'adresse IP du client ne soit généralement pas présente dans les en-têtes HTTP*, les serveurs web peuvent trouver l'adresse IP de l'autre côté de la connexion TCP acheminant la requête HTTP. Par exemple, sur les systèmes Unix, l'appel à la fonction getpeername renvoie l'adresse IP du client de la machine émettrice :

statut = getpeername(tcp_connection_socket,...);

Malheureusement, l’utilisation de l’adresse IP du client pour identifier l’utilisateur présente de nombreuses faiblesses qui limitent son efficacité en tant que technologie d’identification de l’utilisateur :

• Les adresses IP des clients décrivent uniquement l'ordinateur utilisé, et non l'utilisateur. Si plusieurs utilisateurs partagent le même ordinateur, ils seront indiscernables.

• De nombreux fournisseurs de services Internet attribuent dynamiquement des adresses IP aux utilisateurs lorsqu'ils se connectent. Chaque fois qu'ils se connectent, ils obtiennent une adresse différente, de sorte que les serveurs Web ne peuvent pas supposer que les adresses IP identifieront un utilisateur au cours des sessions de connexion.

* Comme nous le verrons plus tard, certains proxys ajoutent un en-tête Client-ip, mais cela ne fait pas partie de la norme HTTP.

Adresse IP du client

• Pour améliorer la sécurité et gérer la rareté des adresses, de nombreux utilisateurs naviguent sur Internet via des pare-feu de traduction d'adresses réseau (NAT). Ces dispositifs NAT masquent les adresses IP des clients réels derrière le pare-feu, convertissant l'adresse IP réelle du client en une adresse IP de pare-feu unique et partagée (avec différents numéros de port).

• Les proxys et passerelles HTTP ouvrent généralement de nouvelles connexions TCP vers le serveur d'origine. Le serveur web verra l'adresse IP du serveur proxy plutôt que celle du client. Certains proxys tentent de contourner ce problème en ajoutant des en-têtes d'extension HTTP spéciaux « Client-ip » ou « X-Forwarded-For » afin de préserver l'adresse IP d'origine (Figure 11-1). Cependant, tous les proxys ne prennent pas en charge ce comportement.

Figure 11-1. Les proxys peuvent ajouter des en-têtes d'extension pour transmettre l'adresse IP d'origine du client.

Certains sites web utilisent encore les adresses IP des clients pour suivre les utilisateurs entre les sessions, mais ils sont peu nombreux. Il existe de nombreux endroits où le ciblage par adresse IP ne fonctionne pas bien.

Certains sites utilisent même les adresses IP client comme mesure de sécurité, ne diffusant les documents qu'aux utilisateurs d'une adresse IP spécifique. Si cette solution peut être efficace sur un intranet, elle est inefficace sur Internet, principalement en raison de la facilité avec laquelle les adresses IP peuvent être falsifiées. La présence de proxys intercepteurs sur le chemin d'accès compromet également ce système. Le chapitre 14 présente des méthodes beaucoup plus efficaces pour contrôler l'accès aux documents privilégiés.

Connexion utilisateur

Plutôt que d'essayer passivement de deviner l'identité d'un utilisateur à partir de son adresse IP, un serveur Web peut explicitement demander à l'utilisateur qui il est en lui demandant de s'authentifier (se connecter) avec un nom d'utilisateur et un mot de passe.

Pour faciliter la connexion aux sites web, HTTP intègre un mécanisme permettant de transmettre les informations d'identification aux sites web, grâce aux en-têtes WWW-Authenticate et Authorization. Une fois connecté, les navigateurs envoient ces informations de connexion à chaque requête adressée au site, les rendant ainsi toujours disponibles. Nous aborderons cette authentification HTTP plus en détail au chapitre 12, mais examinons-la rapidement pour l'instant.

Si un serveur souhaite qu'un utilisateur s'inscrive avant de lui donner accès au site, il peut renvoyer une réponse HTTP 401 « Connexion requise » au navigateur. Ce dernier affichera alors une boîte de dialogue de connexion et fournira les informations dans la requête suivante, via l'en-tête « Authorization ».* Ceci est illustré à la figure 11-2.

Figure 11-2. Enregistrement du nom d'utilisateur à l'aide des en-têtes d'authentification HTTP

Voici ce qui se passe dans cette figure :

• Dans la figure 11-2a, un navigateur effectue une requête depuis le site www.joes-hardware.com.

• Le site ne connaît pas l'identité de l'utilisateur, donc dans la figure 11-2b, le serveur demande une connexion en renvoyant le code de réponse HTTP 401 Connexion requise et

* Pour éviter aux utilisateurs de devoir se connecter pour chaque demande, la plupart des navigateurs mémorisent les informations de connexion d'un site et transmettent les informations de connexion pour chaque demande au site.

Connexion utilisateur

Ajoute l'en-tête WWW-Authenticate. Cela entraîne l'affichage d'une boîte de dialogue de connexion dans le navigateur.

• Une fois que l'utilisateur a saisi un nom d'utilisateur et un mot de passe (pour vérifier son identité), le navigateur répète la requête initiale. Cette fois, il ajoute un en-tête d'autorisation spécifiant le nom d'utilisateur et le mot de passe. Ces derniers sont cryptés afin de les dissimuler aux observateurs occasionnels ou accidentels du réseau.*

• Désormais, le serveur connaît l’identité de l’utilisateur.

• Pour les requêtes ultérieures, le navigateur fournira automatiquement le nom d'utilisateur et le mot de passe enregistrés

lorsqu'ils seront demandés, et les enverra même souvent au site si ce n'est pas demandé. Cela permet de se connecter une fois à un site et de préserver son identité tout au long de la session, le navigateur envoyant l'en-tête d'autorisation comme preuve de son identité à chaque requête au serveur.

Cependant, se connecter à des sites web est fastidieux. En naviguant d'un site à l'autre, Fred devra se connecter à chaque site. Pire encore, il est probable que le pauvre Fred devra mémoriser différents noms d'utilisateur et mots de passe pour chaque site. Son nom d'utilisateur préféré, « fred », aura déjà été choisi par quelqu'un d'autre lorsqu'il visitera plusieurs sites, et certains sites auront des règles différentes concernant la longueur et la composition des noms d'utilisateur et mots de passe. Très vite, Fred abandonnera Internet et retournera regarder Oprah. La section suivante propose une solution à ce problème.

URL en gras

Certains sites web conservent l'identité des utilisateurs en générant des versions spécifiques de chaque URL. Généralement, une URL réelle est étendue en ajoutant des informations d'état au début ou à la fin du chemin d'accès. À mesure que l'utilisateur navigue sur le site, le serveur web génère dynamiquement des hyperliens qui conservent les informations d'état des URL.

Les URL modifiées pour inclure des informations sur l'état de l'utilisateur sont appelées « URL complètes ». Voici quelques exemples d'URL complètes utilisées sur le site e-commerce Amazon.com. Chaque URL est suivie d'un numéro d'identification unique (002-1145265-8016838, dans ce cas) qui permet de suivre l'utilisateur lors de sa navigation sur la boutique.

...

Tous

Cadeaux

Liste de souhaits
 ...

<a href="http://s1.amazon.com/exec/varzea/tg/armed-forces/-//ref=gr_af_/002-1145265-

8016838">Salut nos troupes

Gratuit

Expédition

* Comme nous le verrons au chapitre 14, le nom d'utilisateur et le mot de passe d'authentification HTTP de base peuvent être facilement déchiffrés par quiconque le souhaite avec un minimum d'effort. Des techniques plus sécurisées seront abordées ultérieurement.

Facile

Retours ...

Vous pouvez utiliser des URL lourdes pour lier les transactions HTTP indépendantes avec un serveur web en une seule « session » ou « visite ». Lors de la première visite d'un utilisateur sur le site web, un identifiant unique est généré, ajouté à l'URL de manière reconnaissable par le serveur, qui redirige le client vers cette URL lourde. Chaque fois que le serveur reçoit une requête pour une URL lourde, il peut rechercher tout état incrémentiel associé à cet identifiant utilisateur (panier, profil, etc.) et réécrit tous les hyperliens sortants pour les rendre lourds et conserver l'identifiant utilisateur.

Les URL complètes peuvent être utilisées pour identifier les utilisateurs lorsqu'ils naviguent sur un site. Cependant, cette technologie présente plusieurs problèmes majeurs, notamment :

URL laides

Les URL volumineuses affichées dans le navigateur sont déroutantes pour les nouveaux utilisateurs.

Impossible de partager les URL

Les URL complètes contiennent des informations sur un utilisateur et une session spécifiques. Si vous envoyez cette URL à quelqu'un d'autre, vous risquez de partager par inadvertance vos informations personnelles accumulées.

Interrompt la mise en cache

La génération de versions spécifiques à l'utilisateur de chaque URL signifie qu'il n'y a plus d'URL fréquemment consultées à mettre en cache.

Charge supplémentaire du serveur

Le serveur doit réécrire les pages HTML pour engraisser les URL.

trappes d'évacuation

Il est très facile pour un utilisateur de quitter accidentellement une session d'URL lourde en accédant à un autre site ou en demandant une URL particulière. Les URL lourdes ne fonctionnent que si l'utilisateur suit scrupuleusement les liens pré-modifiés. S'il s'échappe, il risque de

perdre sa progression (éventuellement un panier plein) et devra recommencer.

Non persistant d'une session à l'autre

Toutes les informations sont perdues lorsque l'utilisateur se déconnecte, à moins qu'il ne mette en favoris l'URL spécifique.

Cookies

Les cookies constituent actuellement le meilleur moyen d'identifier les utilisateurs et de permettre des sessions persistantes. Ils ne présentent pas beaucoup des inconvénients des techniques précédentes, mais sont souvent utilisés en complément de ces dernières pour en tirer un avantage supplémentaire. Initialement développés par Netscape, les cookies sont désormais pris en charge par tous les principaux navigateurs.

Les cookies étant importants et définissant de nouveaux en-têtes HTTP, nous allons les explorer plus en détail que les techniques précédentes. La présence de cookies a également un impact sur la mise en cache, et la plupart des caches et navigateurs interdisent la mise en cache de tout contenu contenant des cookies. Les sections suivantes présentent plus de détails.

Types de cookies

Il existe deux grandes catégories de cookies : les cookies de session et les cookies persistants. Un cookie de session est un cookie temporaire qui enregistre les paramètres et les préférences de l'utilisateur lorsqu'il navigue sur un site. Il est supprimé lorsque l'utilisateur quitte son navigateur. Les cookies persistants ont une durée de vie plus longue ; ils sont stockés sur le disque dur et survivent aux fermetures de navigateur et aux redémarrages de l'ordinateur. Les cookies persistants sont souvent utilisés pour conserver un profil de configuration ou un identifiant de connexion pour un site qu'un utilisateur consulte régulièrement.

La seule différence entre les cookies de session et les cookies persistants réside dans leur date d'expiration. Comme nous le verrons plus loin, un cookie est considéré comme un cookie de session si son paramètre « Discard » est défini, ou si aucun paramètre « Expire » ou « Max-Age » n'indique une date d'expiration prolongée.

Comment fonctionnent les cookies

Les cookies sont comme des autocollants « Bonjour, je m'appelle » collés sur les utilisateurs par les serveurs. Lorsqu'un utilisateur visite un site web, celui-ci peut lire tous les autocollants collés sur lui par ce serveur.

Lors de la première visite d'un utilisateur sur un site web, le serveur web ne connaît rien de lui (Figure 11-3a). S'attendant à ce que cet

utilisateur revienne, il souhaite lui associer un cookie unique afin de pouvoir l'identifier ultérieurement. Ce cookie contient une liste arbitraire d'informations nom=valeur et est associé à l'utilisateur via les en-têtes de réponse HTTP (extension) Set-Cookie ou Set-Cookie2.

Les cookies peuvent contenir n'importe quelle information, mais ils contiennent souvent simplement un numéro d'identification unique, généré par le serveur à des fins de suivi. Par exemple, dans la figure 11-3b, le serveur place sur l'utilisateur un cookie portant l'identifiant « 34294 ». Ce numéro permet au serveur de consulter les informations de la base de données qu'il collecte pour ses visiteurs (historique d'achat, adresse, etc.).

Cependant, les cookies ne se limitent pas aux seuls identifiants. De nombreux serveurs web choisissent de conserver les informations directement dans les cookies. Par exemple :

Cookie : nom = Brian Totty ; téléphone = 555-1212

Le navigateur mémorise le contenu des cookies renvoyés par le serveur dans les en-têtes Set-Cookie ou Set-Cookie2, et stocke l'ensemble des cookies dans une base de données de cookies (imaginez une valise avec des autocollants de différents pays). Lorsque l'utilisateur revient sur le même site (Figure 11-3c), le navigateur sélectionne les cookies déposés par le serveur et les lui renvoie dans un en-tête de requête Cookie.

Cookie Jar : État côté client

L'idée de base des cookies est de permettre au navigateur d'accumuler un ensemble d'informations spécifiques au serveur et de renvoyer ces informations au serveur à chaque fois que vous visitez.

Figure 11-3. Placer un cookie sur un utilisateur

Étant donné que le navigateur est responsable du stockage des informations des cookies, ce système est appelé « état côté client ». Le nom officiel de la spécification des cookies est HTTP.

Mécanisme de gestion de l'État.

Cookies de Netscape Navigator

Les navigateurs stockent les cookies de différentes manières. Netscape Navigator les stocke dans un fichier texte unique appelé cookies.txt. Par exemple :

Fichier de cookies HTTP Netscape

http://www.netscape.com/newsref/std/cookie_spec.html # Ceci est un fichier généré ! Ne pas modifier.

#

domaine allh chemin sécurisé expire nom valeur

www.fedex.com FAUX / FAUX 1136109676 cc /us/
.bankofamericaonline.com VRAI / FAUX 1009789256 état CA

.cnn.com VRAI / FAUX 1035069235 SelEdition www secure.eepulse.net FAUX /eePulse FAUX 1007162968 cid %FE%FF%002 www.reformamt.org VRAI /forum FAUX 1033761379 Dernière visite 1003520952 www.reformamt.org VRAI /forum FAUX 1033761379 Nom d'utilisateur Invité ...

Chaque ligne du fichier texte représente un cookie. Il comporte sept champs séparés par des tabulations : domaine

Le domaine du cookie allh

Si tous les hôtes d'un domaine reçoivent le cookie, ou seulement l'hôte spécifique nommé path

Le préfixe du chemin dans le domaine associé au cookie sécurisé

Si nous devons envoyer ce cookie uniquement si nous avons une expiration de connexion SSL

La date d'expiration du cookie en secondes depuis le 1er janvier 1970 00:00:00 GMT nom

Le nom de la valeur de la variable de cookie

La valeur de la variable cookie

Cookies de Microsoft Internet Explorer

Microsoft Internet Explorer stocke les cookies dans des fichiers texte individuels dans le répertoire cache. Vous pouvez parcourir ce répertoire pour afficher les cookies, comme illustré à la figure 11-4. Le format des fichiers cookies d'Internet Explorer est propriétaire, mais de nombreux champs sont faciles à comprendre. Chaque cookie est stocké l'un après l'autre dans le fichier et comporte plusieurs lignes.

La première ligne de chaque cookie du fichier contient le nom de la variable. La ligne suivante contient la valeur de la variable. La troisième ligne contient le domaine et le chemin d'accès. Les lignes restantes contiennent des données propriétaires, incluant probablement des dates et d'autres indicateurs.

Différents cookies pour différents sites

Un navigateur peut contenir des centaines, voire des milliers de cookies, mais ils n'envoient pas tous les cookies à chaque site. En fait, ils n'envoient généralement que deux ou trois cookies par site. Voici pourquoi :

• Déplacer tous ces octets de cookies ralentirait considérablement les performances. Les navigateurs déplaceraient en réalité plus d'octets de cookies que de contenu réel !

• La plupart de ces cookies ne seraient qu'un charabia méconnaissable pour la plupart des sites, car ils contiennent des paires nom/valeur spécifiques au serveur.

• L'envoi de tous les cookies à tous les sites créerait un problème potentiel de confidentialité, les sites auxquels vous ne faites pas confiance obtenant des informations que vous souhaitiez uniquement pour un autre site.

Figure 11-4. Les cookies d'Internet Explorer sont stockés dans des fichiers texte individuels dans le répertoire cache.

En général, un navigateur envoie à un serveur uniquement les cookies générés par ce dernier. Les cookies générés par joes-hardware.com sont envoyés à joes-hardware.com et non à bobs-books.com ou marys-movies.com.

De nombreux sites web font appel à des prestataires tiers pour la gestion de leurs publicités. Ces publicités sont conçues pour apparaître comme faisant partie intégrante du site et utilisent des cookies persistants. Lorsque l'utilisateur consulte un autre site web géré par la même agence publicitaire, le cookie persistant précédemment défini est renvoyé par le navigateur (car les domaines correspondent). Une agence marketing pourrait utiliser cette technique, combinée à l'en-tête Referer, pour potentiellement constituer un ensemble de données exhaustif sur les profils et les habitudes de navigation des utilisateurs. Les navigateurs modernes permettent de configurer les paramètres de confidentialité pour restreindre les cookies tiers.

Attribut de domaine de cookie

Un serveur générant un cookie peut contrôler les sites qui peuvent le voir en ajoutant un attribut Domaine à l'en-tête de réponse Set-Cookie. Par exemple, l'en-tête de réponse HTTP suivant indique au navigateur d'envoyer le cookie user=“mary17” à tout site du domaine .airtravelbargains.com :

Set-cookie : utilisateur = mary17 ; domaine = airtravelbargains.com

Si l'utilisateur visite www.airtravelbargains.com, specials.airtravelbargains.com ou tout autre site se terminant par .airtravelbargains.com, l'en-tête de cookie suivant sera émis :

Cookie : utilisateur = mary17 »

Attribut de chemin de cookie

La spécification des cookies permet même d'associer des cookies à des parties de sites web. Cela se fait grâce à l'attribut Path, qui indique le préfixe du chemin d'URL où chaque cookie est valide.

Par exemple, un serveur web peut être partagé entre deux organisations, chacune disposant de cookies distincts. Le site www.airtravelbargains.com pourrait consacrer une partie de son site web à la location de voitures (par exemple, http://www.airtravelbargains.com/autos/) et utiliser un cookie distinct pour suivre la taille de voiture préférée de l'utilisateur. Un cookie spécifique à la location de voitures pourrait être généré comme suit :

Set-cookie : pref=compact ; domain="airtravelbargains.com" ; path=/autos/

Si l'utilisateur se rend sur http://www.airtravelbargains.com/specials.html, il n'obtiendra que ce cookie :

Cookie : utilisateur = mary17 »

Mais si elle va sur http://www.airtravelbargains.com/autos/cheapo/index.html, elle obtiendra ces deux cookies :

Cookie : utilisateur = mary17 »

Cookie : pref=compact

Les cookies sont des éléments d'état, placés sur le client par les serveurs, gérés par les clients et renvoyés uniquement aux sites appropriés. Examinons plus en détail la technologie et les normes des cookies.

Ingrédients des biscuits

Il existe deux versions différentes de spécifications de cookies : les cookies de version 0 (parfois appelés « cookies Netscape ») et les cookies de version 1 (« RFC 2965 »). Les cookies de version 1 sont une extension moins répandue des cookies de version 0.

Ni la version 0 ni la version 1 de la spécification des cookies ne sont documentées dans la spécification HTTP/1.1. Deux documents complémentaires principaux décrivent parfaitement l'utilisation des cookies, résumés dans le tableau 11-2.

Tableau 11-2. Spécifications des cookies

Titre Description Localisation

État du client persistant : Cookies HTTP Norme de cookie Netscape d'origine http://home.netscape.com/newsref/std/cookie_spec.html

RFC 2965 : Gestion de l'état HTTP

Mécanisme Norme de cookie d'octobre 2000, obsolète RFC 2109 http://www.ietf.org/rfc/rfc2965.txt

Version 0 (Netscape) Cookies

La spécification initiale des cookies a été définie par Netscape. Ces cookies de « version 0 » définissaient l'en-tête de réponse Set-Cookie, l'en-tête de requête Cookie et les champs disponibles pour le contrôle des cookies. Les cookies de la version 0 se présentent ainsi :

Set-Cookie : nom=valeur [; expire=date] [; chemin=chemin] [; domaine=domaine] [; sécurisé] Cookie : nom1=valeur1 [; nom2=valeur2] ...

Version 0 Set-Cookie header

L'en-tête Set-Cookie contient un nom et une valeur de cookie obligatoires. Il peut être suivi d'attributs de cookie facultatifs, séparés par des points-virgules. Les champs Set-Cookie sont décrits dans le tableau 11-3.

Tableau 11-3. Version 0 (Netscape) Attributs Set-Cookie

Attribut Set-Cookie Description et exemples

NOM=VALEUR Obligatoire. NOM et VALEUR sont des séquences de caractères, excluant le point-virgule, la virgule, le signe égal et les espaces, sauf s'ils sont entre guillemets. Le serveur web peut créer n'importe quel élément.

Association NAME=VALUE, qui sera renvoyée au serveur Web lors des visites ultérieures sur le site.

Set-Cookie : client = Mary

Expiration (facultatif). Cet attribut spécifie une date définissant la durée de validité du cookie. Une fois la date d'expiration atteinte, le cookie ne sera plus stocké ni distribué. La date est au format suivant :

Jour de la semaine, JJ-Lun-AA HH:MM:SS GMT

Le seul fuseau horaire légal est GMT, et les séparateurs de date doivent être des tirets. Si « Expire » n'est pas spécifié, le cookie expirera à la fin de la session de l'utilisateur.

Set-Cookie : foo=bar ; expire le mercredi 9 novembre 1999 à 23h12 :40 GMT

Domaine facultatif. Un navigateur envoie le cookie uniquement aux noms d'hôtes du serveur du domaine spécifié. Cela permet aux serveurs de limiter les cookies à certains domaines. Le domaine « acme.com » correspond aux noms d'hôtes « anvil.acme.com » et « shipping.crate.acme.com », mais pas à « www.cnn.com ».

Seuls les hôtes du domaine spécifié peuvent définir un cookie pour un domaine. Ces domaines doivent comporter au moins deux ou trois points afin d'éviter les domaines de type « .com », « .edu » et « va.us ». Tout domaine appartenant à l'ensemble fixe de domaines de premier niveau spéciaux répertoriés ici ne nécessite que deux points. Tout autre domaine en nécessite au moins trois. Les domaines de premier niveau spéciaux sont : .com, .edu, .net, .org, .gov, .mil, .int, .biz, .info, .name, .museum, .coop, .aero et .pro.

Si le domaine n'est pas spécifié, il s'agit par défaut du nom d'hôte du serveur qui a généré la réponse Set-Cookie.

Set-Cookie : EXPÉDITION = FEDEX ; domaine = joes-hardware.com

Chemin facultatif. Cet attribut permet d'attribuer des cookies à des documents spécifiques sur un serveur. Si l'attribut Chemin est un préfixe d'URL, un cookie peut y être attaché. Le chemin « /foo » correspond à « /foobar » et « /foo/bar.html ». Le chemin « / » correspond à tout le domaine.

Si le chemin n'est pas spécifié, il est défini sur le chemin de l'URL qui a généré la réponse Set-Cookie.

Set-Cookie : lastorder=00183 ; chemin=/orders

Sécurisé (facultatif). Si cet attribut est inclus, un cookie sera envoyé uniquement si HTTP utilise une connexion sécurisée SSL.

Set-Cookie : private_id=519 ; sécurisé

Version 0 En-tête de cookie

Lorsqu'un client envoie des requêtes, il inclut tous les cookies non expirés correspondant au domaine, au chemin d'accès et aux filtres sécurisés du site. Tous les cookies sont regroupés dans un en-tête de cookie :

Cookie : session-id=002-1145265-8016838 ; session-id-time=1007884800

Version 1 (RFC 2965) Cookies

Une version étendue des cookies est définie dans la RFC 2965 (anciennement RFC 2109). Cette version 1 introduit les en-têtes Set-Cookie2 et Cookie2, mais elle interagit également avec le système de la version 0.

La norme RFC 2965 relative aux cookies est un peu plus complexe que la norme Netscape originale et n'est pas encore entièrement prise en charge. Les principaux changements apportés aux cookies RFC 2965 sont les suivants :

• Associer un texte descriptif à chaque cookie pour expliquer son objectif

• Prise en charge de la destruction forcée des cookies à la sortie du navigateur, quelle que soit leur expiration

• Vieillissement Max-Age des cookies en secondes relatives, au lieu de dates absolues

• Possibilité de contrôler les cookies par le numéro de port de l'URL, pas seulement par le domaine et le chemin

• L'en-tête du cookie renvoie les filtres de domaine, de port et de chemin (le cas échéant)

• Numéro de version pour l'interopérabilité

• Préfixe $ dans l'en-tête du cookie pour distinguer les mots-clés supplémentaires des noms d'utilisateur La syntaxe du cookie de la version 1 est la suivante :

set-cookie = "Set-Cookie2:" cookies cookies = 1#cookie cookie = NOM "=" VALEUR *(";" set-cookie-av)

NOM = attr

VALEUR = valeur

set-cookie-av = "Commentaire" "=" valeur

| "CommentURL" "=" <"> http_URL <">

| « Jeter »

| "Domaine" "=" valeur

| "Max-Age" "=" valeur

| "Chemin" "=" valeur

| "Port" ["=" <"> liste de ports <">]

| "Sécurisé"

| "Version" "=" 1*DIGIT portlist = 1#portnum portnum = 1*DIGIT

cookie = "Cookie:" cookie-version 1*((";" | ",") cookie-value) cookie-value = NAME "=" VALUE [";" path] [";" domain] [";" port] cookie-version = "$Version" "=" value

NOM = attr

VALEUR = valeur

chemin = "$Chemin" "=" valeur domaine = "$Domaine" "=" valeur port = "$Port" ["=" <"> valeur <">] cookie2 = "Cookie2:" cookie-version

Version 1 En-tête Set-Cookie2

La norme relative aux cookies version 1 propose davantage d'attributs que la norme Netscape. Le tableau 11-4 fournit un résumé rapide des attributs. Pour une explication plus détaillée, reportez-vous à la RFC 2965.

Tableau 11-4. Version 1 (RFC 2965) Attributs Set-Cookie2

Attribut Set-Cookie2 Description et exemples

NOM=VALEUR Obligatoire. Le serveur web peut créer n'importe quelle association NOM=VALEUR, qui lui sera renvoyée lors des prochaines visites sur le site. Le nom ne doit pas commencer par « $ », car ce caractère est réservé.

Version obligatoire. La valeur de cet attribut est un entier correspondant à la version de la spécification du cookie. La RFC 2965 correspond à la version 1.

Set-Cookie2 : Part="Rocket_Launcher_0001"; Version="1"

Commentaire facultatif. Cet attribut documente l'utilisation prévue du cookie par le serveur. L'utilisateur peut consulter cette politique pour décider d'autoriser ou non une session avec ce cookie. La valeur doit être encodée en UTF-8.

CommentURL (facultatif). Cet attribut fournit une URL pointant vers une documentation détaillée sur l'objectif et la politique d'un cookie. L'utilisateur peut consulter cette politique pour décider d'autoriser ou non une session avec ce cookie.

Attribut facultatif : Supprimer. Si cet attribut est présent, il indique au client de supprimer le cookie à la fin du programme client.

Domaine facultatif. Un navigateur envoie le cookie uniquement aux noms d'hôtes du serveur du domaine spécifié. Cela permet aux serveurs de limiter les cookies à certains domaines. Le domaine « acme.com » correspond aux noms d'hôtes « anvil.acme.com » et « shipping.crate.acme.com », mais pas à « www.cnn.com ». Les règles de correspondance des domaines sont globalement les mêmes que pour les cookies Netscape, avec quelques règles supplémentaires. Consultez la RFC 2965 pour plus de détails.

Max-Age (facultatif). La valeur de cet attribut est un entier définissant la durée de vie du cookie en secondes.

Les clients doivent calculer l'âge du cookie conformément aux règles de calcul HTTP/1.1. Lorsque l'âge d'un cookie dépasse l'âge maximal, le client doit le supprimer. Une valeur de zéro signifie que le cookie portant ce nom doit être supprimé immédiatement.

Chemin facultatif. Cet attribut permet d'attribuer des cookies à des documents spécifiques sur un serveur. Si l'attribut Chemin est un préfixe d'un chemin d'URL, un cookie peut être joint. Le chemin

« /foo » correspond à « /foobar » et « /foo/bar.html ». Le chemin « / » correspond à tout le domaine. Si le chemin n'est pas spécifié, il est défini sur le chemin de l'URL ayant généré la réponse Set-Cookie.

Port (facultatif). Cet attribut peut être utilisé seul comme mot-clé ou inclure une liste de ports séparés par des virgules auxquels un cookie peut être appliqué. Si une liste de ports est disponible, le cookie ne peut être servi qu'aux serveurs dont les ports correspondent à un port de la liste. Si le mot-clé Port est fourni isolément, le cookie ne peut être servi qu'au numéro de port du serveur de réponse actuel.

Set-Cookie2 : foo="bar"; Version="1"; Port="80,81,8080"

Set-Cookie2 : foo="bar"; Version="1"; Port

Sécurisé (facultatif). Si cet attribut est inclus, un cookie sera envoyé uniquement si HTTP utilise une connexion sécurisée SSL.

Version 1 En-tête de cookie

Les cookies de la version 1 contiennent des informations supplémentaires sur chaque cookie livré, décrivant les filtres qu'il a passés. Chaque cookie correspondant doit inclure les attributs Domaine, Port ou Chemin d'accès des en-têtes Set-Cookie2 correspondants.

Par exemple, supposons que le client ait reçu ces cinq réponses Set-Cookie2 dans le passé à partir du site Web www.joes-hardware.com :

Set-Cookie2 : ID = « 29046 » ; Domaine = « .joes-hardware.com »

Set-Cookie2 : couleur = bleu

Set-Cookie2 : support-pref="L2"; Domaine="customer-care.joes-hardware.com"

Set-Cookie2 : Coupon = « hammer027 » ; Version = 1 ;
Chemin = /outils ;

Set-Cookie2 : Coupon="handvac103"; Version="1";
Chemin="/tools/cordless"

Si le client fait une autre demande pour le chemin /tools/cordless/specials.html, il transmettra un long en-tête Cookie2 comme celui-ci :

Cookie : $Version="1";

ID="29046"; $Domain=".joes-hardware.com"; color="blue";

Coupon="hammer027"; $Path="/tools";

Coupon="handvac103"; $Path="/tools/cordless"

Notez que tous les cookies correspondants sont livrés avec leurs filtres Set-Cookie2 et que les mots-clés réservés commencent par un signe dollar ($).

En-tête Cookie2 version 1 et négociation de version

L'en-tête de requête Cookie2 permet de négocier l'interopérabilité entre les clients et les serveurs qui comprennent différentes versions de la spécification des cookies. L'en-tête Cookie2 informe le serveur que l'agent utilisateur comprend les nouveaux cookies et fournit la version de la norme de cookie prise en charge (il aurait été plus logique de l'appeler Cookie-Version) :

Cookie2 : $Version="1"

Si le serveur reconnaît les cookies de nouvelle génération, il reconnaît l'en-tête Cookie2 et doit envoyer les en-têtes de réponse Set-Cookie2 (plutôt que Set-Cookie). Si un client reçoit à la fois un en-tête Set-Cookie et un en-tête Set-Cookie2 pour le même cookie, il ignore l'ancien en-tête Set-Cookie.

Si un client prend en charge les cookies Version 0 et Version 1, mais reçoit du serveur un en-tête SetCookie Version 0, il doit envoyer des cookies avec l'en-tête Cookie Version 0. Cependant, le client doit également envoyer Cookie2: $Version=“1” pour indiquer au serveur qu'il peut effectuer la mise à niveau.

Cookies et suivi de session

Les cookies permettent de suivre les utilisateurs lors de leurs multiples transactions sur un site web. Les sites de e-commerce utilisent des cookies de session pour suivre les paniers d'achat des utilisateurs lors de leur navigation. Prenons l'exemple du célèbre site de vente en ligne Amazon.com.

Lorsque vous saisissez http://www.amazon.com dans votre navigateur, vous démarrez une chaîne de transactions où le serveur Web joint des informations d'identification via une série de redirections, de réécritures d'URL et de paramètres de cookies.

La figure 11-5 montre une séquence de transaction capturée à partir d'une visite sur Amazon.com :

• Figure 11-5a — Le navigateur demande la page racine d’Amazon.com pour la première fois.

• Figure 11-5b — Le serveur redirige le client vers une URL pour le logiciel de commerce électronique.

• Figure 11-5c — Le client fait une demande à l’URL redirigée.

Figure 11-5d — Le serveur ajoute deux cookies de session à la réponse et redirige l'utilisateur vers une autre URL. Le client effectuera donc une nouvelle requête avec ces cookies. Cette nouvelle URL est une URL

complète, ce qui signifie qu'un état y est intégré. Si les cookies sont désactivés sur le client, une identification de base est possible tant que l'utilisateur suit les liens de l'URL complète générée par Amazon.com et ne quitte pas le site.

• Figure 11-5e — Le client demande la nouvelle URL, mais transmet désormais les deux cookies joints.

• Figure 11-5f — Le serveur redirige vers la page home.html et joint deux autres cookies.

• Figure 11-5g — Le client récupère la page home.html et transmet les quatre cookies.

• Figure 11-5h — Le serveur renvoie le contenu.

Cookies et mise en cache

Soyez prudent lors de la mise en cache de documents impliquant des transactions de cookies. Vous ne souhaitez pas attribuer à un utilisateur le cookie d'un ancien utilisateur ou, pire, lui montrer le contenu du document personnalisé d'un autre utilisateur.

Les règles relatives aux cookies et à la mise en cache ne sont pas clairement établies. Voici quelques principes directeurs pour gérer les caches :

Marquer les documents comme non cachables s'ils sont

Le propriétaire du document est le mieux placé pour savoir si un document est non cachable. Indiquez explicitement que les documents ne peuvent pas être mis en cache s'ils le sont. Utilisez notamment Cache-Control: nocache=“Set-Cookie” si le document est cachable, à l'exception de l'en-tête Set-Cookie. L'autre pratique, plus générale, consistant à utiliser Cache-Control: public pour les documents cachables favorise les économies de bande passante sur le Web.

Soyez prudent lors de la mise en cache des en-têtes Set-Cookie

Si une réponse comporte un en-tête Set-Cookie, vous pouvez mettre le corps en cache (sauf indication contraire), mais soyez particulièrement vigilant quant à la mise en cache de l'en-tête Set-Cookie. Envoyer le même en-tête Set-Cookie à plusieurs utilisateurs peut compromettre le ciblage.

Certains caches suppriment l'en-tête Set-Cookie avant de stocker une réponse dans le cache, mais cela peut également causer des problèmes, car les clients servis à partir du cache ne

Figure 11-5. Le site web Amazon.com utilise des cookies de session pour suivre les utilisateurs.

Les cookies ne sont plus appliqués comme ils le feraient normalement sans le cache. Cette situation peut être améliorée en forçant le cache à revalider chaque requête auprès du serveur d'origine et en fusionnant les en-têtes Set-Cookie renvoyés avec la réponse du client. Le serveur d'origine peut imposer ces revalidations en ajoutant cet en-tête à la copie mise en cache :

Cache-Control : doit être revalidé, max-age=0

Les caches plus conservateurs peuvent refuser de mettre en cache toute réponse comportant un en-tête SetCookie, même si son contenu peut être mis en cache. Certains caches autorisent la mise en cache des images SetCookie, mais pas du texte.

Soyez prudent avec les requêtes contenant des en-têtes de cookies

Lorsqu'une requête arrive avec un en-tête Cookie, elle indique que le contenu obtenu pourrait être personnalisé. Le contenu personnalisé doit être signalé comme non cachable, mais certains serveurs peuvent, par erreur, ne pas le signaler comme tel.

Les caches conservateurs peuvent choisir de ne pas mettre en cache les documents reçus en réponse à une requête contenant un en-tête Cookie. De plus, certains caches autorisent la mise en cache des images contenant des Cookies, mais pas du texte. La politique la plus répandue consiste à mettre en cache les images contenant des en-têtes Cookie, avec un délai d'expiration fixé à zéro, ce qui force une revalidation à chaque fois.

Cookies, sécurité et confidentialité

Les cookies ne sont pas considérés comme présentant un risque majeur pour la sécurité, car ils peuvent être désactivés et une grande partie du suivi peut être effectuée par l'analyse des journaux ou par d'autres moyens. En effet, en fournissant une méthode standardisée et rigoureuse de conservation des informations personnelles dans des bases de données distantes et en utilisant des cookies anonymes comme clés, la fréquence de communication des données sensibles du client au serveur peut être réduite.

réduit.

Il convient néanmoins de rester prudent en matière de confidentialité et de suivi des utilisateurs, car le risque d'abus est toujours présent. Les abus les plus fréquents proviennent de sites web tiers utilisant des cookies persistants pour suivre les utilisateurs. Cette pratique, combinée aux adresses IP et aux informations de l'en-tête Referer, a permis à ces agences marketing de créer des profils d'utilisateurs et des habitudes de navigation relativement précis.

Malgré toute la publicité négative, la sagesse conventionnelle veut que la gestion des sessions et la commodité transactionnelle des cookies

l'emportent sur la plupart des risques, si vous faites preuve de prudence quant aux personnes à qui vous fournissez des informations personnelles et si vous examinez la confidentialité des sites.

politiques.

Le Computer Incident Advisory Capability (qui fait partie du Département de l'Énergie des États-Unis) a rédigé une évaluation des dangers surreprésentés des cookies en 1998. Voici un extrait de ce rapport :

CIAC I-034 : Cookies Internet

(http://www.ciac.org/ciac/bulletins/i-034.shtml)

PROBLÈME:

Les cookies sont de courts fragments de données utilisés par les serveurs web pour identifier les utilisateurs. Les concepts et rumeurs répandus sur les fonctions d'un cookie ont atteint des proportions quasi mystiques, effrayant les utilisateurs et inquiétant leurs responsables.

ÉVALUATION DE LA VULNÉRABILITÉ :

La vulnérabilité des systèmes aux dommages ou à l'espionnage causés par les cookies de navigateur web est quasiment inexistante. Les cookies peuvent seulement indiquer à un serveur web votre précédente visite et peuvent se transmettre de courtes informations (comme un numéro d'utilisateur) lors de votre prochaine visite. La plupart des cookies ne durent que jusqu'à la fermeture de votre navigateur, puis sont détruits. Un deuxième type de cookie, appelé cookie persistant, a une date d'expiration et est stocké sur votre disque dur jusqu'à cette date. Un cookie persistant permet de suivre les habitudes de navigation d'un utilisateur en l'identifiant à chaque visite. Les informations sur votre provenance et les pages web consultées sont déjà présentes dans les fichiers journaux d'un serveur web et peuvent également être utilisées pour suivre les habitudes de navigation des utilisateurs. Les cookies simplifient simplement la tâche.

Pour plus d'informations

Voici quelques sources utiles supplémentaires pour obtenir des informations supplémentaires sur les cookies :

Cookies

Simon St.Laurent, McGraw-Hill.

http://www.ietf.org/rfc/rfc2965.txt

RFC 2965, « Mécanisme de gestion de l’état HTTP » (obsolète RFC 2109).

http://www.ietf.org/rfc/rfc2964.txt

RFC 2964, « Utilisation de la gestion d'état HTTP ».
http://home.netscape.com/newsref/std/cookie_spec.html

Ce document classique de Netscape, « Persistent Client State : HTTP Cookies », décrit la forme originale des cookies HTTP qui sont encore couramment utilisés aujourd'hui.

Chapitre 12Ceci est le titre du livre CHAPITRE 12

Authentification de base

Des millions de personnes utilisent le Web pour effectuer des transactions privées et accéder à des données privées. Le Web facilite grandement l'accès à ces informations, mais cette simplicité ne suffit pas. Nous devons avoir l'assurance que nos données sensibles et nos transactions privilégiées sont accessibles. Toutes les informations ne sont pas destinées au grand public.

Nous devons être certains que des utilisateurs non autorisés ne peuvent pas consulter nos profils de voyage en ligne ni publier de documents sur nos sites web sans notre consentement. Nous devons nous assurer que nos documents de planification d'entreprise les plus sensibles ne soient pas accessibles à des membres non autorisés et potentiellement sans scrupules de notre organisation. Et nous devons être certains que nos communications personnelles en ligne avec nos enfants, nos conjoints et nos amours secrets se déroulent dans un minimum de confidentialité.

Les serveurs doivent pouvoir identifier un utilisateur. Une fois identifié, le serveur peut décider des transactions et des ressources auxquelles il peut accéder. L'authentification consiste à prouver son identité ; généralement, on s'authentifie en fournissant un nom d'utilisateur et un mot de passe secret. HTTP propose une fonctionnalité native d'authentification HTTP. Bien qu'il soit tout à fait possible de déployer sa propre fonctionnalité d'authentification en complément des formulaires et des cookies HTTP, dans de nombreuses situations, l'authentification native HTTP est idéale.

bien.

Ce chapitre explique l'authentification HTTP et se penche sur la forme la plus courante d'authentification HTTP : l'authentification de base. Le chapitre suivant présente une technique plus puissante appelée authentification Digest.

Authentification

L'authentification consiste à prouver votre identité. Présenter une pièce d'identité avec photo, comme un passeport ou un permis de

conduire, prouve que vous êtes bien la personne que vous prétendez être. Saisir un code PIN dans un distributeur automatique ou saisir un mot de passe secret dans une boîte de dialogue d'ordinateur prouve également que vous êtes bien la personne que vous prétendez être.

Aucun de ces stratagèmes n'est infaillible. Les mots de passe peuvent être devinés ou entendus, et les cartes d'identité peuvent être volées ou falsifiées. Mais chaque preuve à l'appui contribue à établir une confiance raisonnable quant à votre identité.

Cadre d'authentification par défi/réponse HTTP

HTTP fournit un framework natif de type « question/réponse » pour faciliter l'authentification des utilisateurs. Le modèle d'authentification HTTP est illustré à la figure 12-1.

Figure 12-1. Authentification simplifiée par défi/réponse

Chaque fois qu'une application Web reçoit un message de requête HTTP, au lieu d'agir sur la requête, le serveur peut répondre par un « défi d'authentification », mettant l'utilisateur au défi de démontrer qui il est en fournissant des informations secrètes.

L'utilisateur doit joindre ses identifiants secrets (nom d'utilisateur et mot de passe) lorsqu'il répète la requête. Si les identifiants ne correspondent pas, le serveur peut relancer le client ou générer une erreur. Si les identifiants correspondent, la requête se termine normalement.

Protocoles d'authentification et en-têtes

HTTP fournit un cadre extensible pour différents protocoles d'authentification, grâce à un ensemble d'en-têtes de contrôle personnalisables. Le format et le contenu des en-têtes répertoriés dans le tableau 12-1 varient selon le protocole d'authentification. Le protocole d'authentification est également spécifié dans les en-têtes d'authentification HTTP.

HTTP définit deux protocoles d'authentification officiels : l'authentification de base et l'authentification par résumé. À l'avenir, chacun sera libre de concevoir de nouveaux protocoles utilisant le framework challenge/réponse de HTTP. La suite de ce chapitre explique l'authentification de base. Voir le chapitre 13 pour plus de détails sur l'authentification par résumé.

Tableau 12-1. Quatre phases d'authentification

Description des en-têtes de phase Méthode/État

Requête La première requête n'a pas d'authentification. GET

Défi WWW-Authenticate Le serveur rejette la requête avec un statut 401, indiquant que l'utilisateur doit fournir son nom d'utilisateur et son mot de passe.

Étant donné que le serveur peut comporter différentes zones, chacune dotée de son propre mot de passe, il décrit la zone de protection dans l'en-tête WWW-Authenticate. L'algorithme d'authentification est également spécifié dans cet en-tête. Erreur 401 : Non autorisé

Autorisation Autorisation Le client réessaye la demande, mais cette fois en joignant un en-tête d'autorisation spécifiant l'algorithme d'authentification, le nom d'utilisateur et le mot de passe.

Informations d'authentification réussies : si les informations d'identification sont correctes, le serveur renvoie le document. Certains algorithmes d'autorisation renvoient des informations supplémentaires sur la session d'autorisation dans l'en-tête facultatif « Authentication-Info ». 200 OK

Pour rendre cela concret, regardons la figure 12-2.

Figure 12-2. Exemple d'authentification de base

Lorsqu'un serveur défie un utilisateur, il renvoie une réponse 401 Non autorisé et décrit comment et où s'authentifier dans l'en-tête WWW-Authenticate

(Figure 12-2b).

Authentification

Lorsqu'un client autorise le serveur à continuer, il renvoie la demande mais joint un mot de passe codé et d'autres paramètres d'authentification dans un en-tête d'autorisation (Figure 12-2c).

Lorsqu'une demande autorisée est exécutée avec succès, le serveur renvoie un code d'état normal (par exemple, 200 OK) et, pour les algorithmes d'authentification avancés, peut joindre des informations supplémentaires dans un en-tête Authentication-Info (Figure 12-2d).

Domaines de sécurité

Avant d'aborder les détails de l'authentification de base, nous devons expliquer comment HTTP permet aux serveurs d'associer différents droits d'accès à différentes ressources. Vous avez peut-être remarqué que le défi WWW-Authenticate de la figure 12-2b incluait une directive « realm ». Les serveurs Web regroupent les documents protégés en domaines de sécurité. Chaque domaine de sécurité peut avoir différents groupes d'utilisateurs autorisés.

Par exemple, supposons qu'un serveur web dispose de deux domaines de sécurité : un pour les informations financières de l'entreprise et un

autre pour les documents familiaux personnels (voir figure 12-3). Chaque utilisateur aura un accès différent à ces domaines. Le PDG de votre entreprise devrait probablement avoir accès aux prévisions de ventes, mais vous ne lui donnerez peut-être pas accès à vos photos de vacances en famille !

Figure 12-3. Domaines de sécurité d'un serveur Web

Voici un défi d'authentification de base hypothétique, avec un domaine spécifié :

HTTP/1.0 401 Non autorisé

WWW-Authenticate : Domaine de base = « Finances de l'entreprise »

Un domaine doit avoir un nom de chaîne descriptif, comme « Finances de l'entreprise », pour aider l'utilisateur à comprendre le nom d'utilisateur et le mot de passe à utiliser. Il peut également être utile d'indiquer le nom d'hôte du serveur dans le nom du domaine, par exemple « comité-exécutif@grandeentreprise.com ».

Authentification de base

L'authentification de base est le protocole d'authentification HTTP le plus répandu. Presque tous les principaux clients et serveurs implémentent l'authentification de base. Initialement décrite dans la spécification HTTP/1.0, elle a depuis été déplacée dans la RFC 2617, qui détaille l'authentification HTTP.

Lors de l'authentification de base, un serveur web peut refuser une transaction en demandant au client un nom d'utilisateur et un mot de passe valides. Le serveur lance la demande d'authentification en renvoyant un code d'état 401 au lieu de 200 et en spécifiant le domaine de sécurité auquel il accède avec l'en-tête de réponse WWW-Authenticate. Lorsque le navigateur reçoit la demande, il ouvre une boîte de dialogue demandant le nom d'utilisateur et le mot de passe de ce domaine. Ces derniers sont renvoyés au serveur dans un format légèrement brouillé, dans un en-tête de requête d'autorisation.

Exemple d'authentification de base

La figure 12-2, plus tôt dans ce chapitre, a montré un exemple détaillé d'authentification de base :

• Dans la figure 12-2a, un utilisateur demande la photo de famille personnelle /family/jeff.jpg.

• Dans la figure 12-2b, le serveur renvoie une erreur 401 « Authentification requise » pour la photo de famille, accompagnée de l'en-tête WWW-Authenticate. Cet en-tête demande une authentification de base pour le domaine « Famille ».

• Dans la figure 12-2c, le navigateur reçoit le défi 401 et ouvre une boîte de dialogue demandant le nom d'utilisateur et le mot de passe du domaine Famille. Lorsque l'utilisateur saisit le nom d'utilisateur et le mot de passe, le navigateur les relie par un deux-points, les encode en une représentation brouillée en base 64 (voir la section suivante) et les renvoie dans l'en-tête d'autorisation.

• Dans la figure 12-2d, le serveur décode le nom d'utilisateur et le mot de passe, vérifie qu'ils sont corrects et renvoie le document demandé dans un message HTTP 200 OK.

Les en-têtes d'authentification de base HTTP WWW-Authenticate et Authorization sont résumés dans le tableau 12-2.

Tableau 12-2. En-têtes d'authentification de base

Syntaxe et description de l'en-tête de défi/réponse

Défi (serveur vers client) : Il peut y avoir différents mots de passe pour différentes parties du site. Le domaine est une chaîne entre guillemets qui nomme l'ensemble des documents demandés, afin que l'utilisateur sache quel mot de passe utiliser.

WWW-Authenticate : royaume de base = royaume-cité

Réponse (client vers serveur) Le nom d'utilisateur et le mot de passe sont reliés par deux points (:) puis convertis en codage base 64, ce qui facilite un peu l'inclusion de caractères internationaux dans les noms d'utilisateur et les mots de passe et réduit la probabilité qu'un examen superficiel produise des noms d'utilisateur et des mots de passe lors de l'observation du trafic réseau.

Autorisation : nom d'utilisateur et mot de passe de base base64

Authentification de base

Notez que le protocole d’authentification de base n’utilise pas l’en-tête AuthenticationInfo que nous avons montré dans le tableau 12-1.

Codage du nom d'utilisateur/mot de passe en base 64

L'authentification HTTP de base regroupe le nom d'utilisateur et le mot de passe (séparés par deux points) et les encode en base 64. Si vous ignorez ce qu'est le codage en base 64, pas d'inquiétude. Vous n'avez pas besoin d'en savoir beaucoup, et si vous êtes curieux, vous trouverez toutes les informations à ce sujet dans l'annexe E. En résumé, le codage en base 64 prend une séquence d'octets de 8 bits et la décompose en blocs de 6 bits. Chaque bloc de 6 bits permet de choisir un caractère dans un alphabet spécial de 64 caractères, composé principalement de lettres et de chiffres.

La figure 12-4 montre un exemple d'utilisation du codage en base 64 pour l'authentification de base. Ici, le nom d'utilisateur est « brian-totty » et le mot de passe « Ow ! ». Le navigateur relie le nom

d'utilisateur et le mot de passe par deux points, ce qui produit la chaîne condensée « brian-totty:Ow ! ». Cette chaîne est ensuite codée en base 64 pour former la chaîne suivante :
« YnJpYW4tdG90dHk6T3ch ».

Figure 12-4. Génération d'un en-tête d'autorisation de base à partir du nom d'utilisateur et du mot de passe

Le codage Base-64 a été inventé pour convertir temporairement des chaînes de données binaires, textuelles et de caractères internationaux (ce qui posait problème sur certains systèmes) en un alphabet portable pour la transmission. Les chaînes d'origine pouvaient ensuite être décodées à distance sans risque de corruption de transmission.

Le codage en base 64 peut être utile pour les noms d'utilisateur et les mots de passe contenant des caractères internationaux ou d'autres caractères interdits dans les en-têtes HTTP (tels que les guillemets, les deux-points et les retours chariot). De plus, comme le codage en base 64 brouille facilement le nom d'utilisateur et le mot de passe, il peut empêcher les administrateurs de consulter accidentellement les noms d'utilisateur et les mots de passe lors de l'administration des serveurs et des réseaux.

Authentification proxy

L'authentification peut également être effectuée par des serveurs proxy intermédiaires. Certaines organisations utilisent des serveurs proxy pour authentifier les utilisateurs avant de les autoriser à accéder aux serveurs, aux réseaux locaux ou aux réseaux sans fil. Les serveurs proxy peuvent constituer un moyen pratique d'assurer un contrôle d'accès unifié aux ressources d'une organisation, car les politiques d'accès peuvent être administrées de manière centralisée sur le serveur proxy. La première étape de ce processus consiste à établir l'identité via l'authentification proxy.

Les étapes de l'authentification proxy sont identiques à celles de l'identification du serveur web. Cependant, les en-têtes et les codes d'état sont différents. Le tableau 12-3 compare les codes d'état et les en-têtes utilisés pour l'authentification du serveur web et du proxy.

Tableau 12-3. Authentification serveur Web ou proxy

Serveur Web Serveur proxy

Code d'état non autorisé : 401 Code d'état non autorisé : 407

WWW-Authentifier Proxy-Authentifier

Autorisation Proxy-Autorisation

Informations d'authentification Informations d'authentification proxy

Les failles de sécurité de l'authentification de base

L'authentification de base est simple et pratique, mais elle n'est pas sécurisée. Elle ne doit être utilisée que pour empêcher tout accès non intentionnel par des tiers non malveillants ou en combinaison avec une technologie de chiffrement telle que SSL.

Tenez compte des failles de sécurité suivantes :

1. L'authentification de base envoie le nom d'utilisateur et le mot de passe sur le réseau sous une forme facilement déchiffrable. En effet, le mot de passe secret est envoyé en clair, accessible à tous. Le codage en base 64 masque le nom d'utilisateur et le mot de passe, réduisant ainsi le risque que des tiers non identifiés les récupèrent par observation accidentelle du réseau. Cependant, avec un nom d'utilisateur et un mot de passe codés en base 64, le décodage peut être effectué facilement en inversant le processus de codage. Le décodage peut même être réalisé en quelques secondes, à la main, avec un crayon et du papier ! Les mots de passe codés en base 64 sont effectivement envoyés « en clair ». Partez du principe que des tiers motivés intercepteront les noms d'utilisateur et les mots de passe envoyés par l'authentification de base. Si cela vous préoccupe, envoyez toutes vos transactions HTTP via des canaux chiffrés SSL ou utilisez un protocole d'authentification plus sécurisé, tel que

authentification digest.

Les failles de sécurité de l'authentification de base

2. Même si le mot de passe secret était codé selon un schéma plus complexe à décoder, un tiers pourrait récupérer le nom d'utilisateur et le mot de passe brouillés et les retransmettre indéfiniment aux serveurs d'origine pour obtenir l'accès. Aucun effort n'est déployé pour empêcher ces attaques par rejeu.

3. Même si l'authentification de base est utilisée pour des applications non critiques, comme le contrôle d'accès à l'intranet d'entreprise ou la personnalisation du contenu, les comportements sociaux rendent cette pratique dangereuse. De nombreux utilisateurs, submergés par une multitude de services protégés par mot de passe, partagent leurs noms d'utilisateur et leurs mots de passe. Un individu malin et malveillant peut, par exemple, récupérer un nom d'utilisateur et un mot de passe en clair sur un site de messagerie gratuit et découvrir que ces mêmes identifiants permettent d'accéder à des sites bancaires en ligne critiques !

4. L'authentification de base n'offre aucune protection contre les proxys ou les intermédiaires qui agissent comme des intermédiaires, laissant les en-têtes d'authentification intacts mais modifiant le reste du message pour changer radicalement la nature de la transaction.

5. L'authentification de base est vulnérable à l'usurpation d'identité par des serveurs contrefaits. Si un utilisateur est amené à croire qu'il se connecte à un hôte valide protégé par l'authentification de base alors qu'il se connecte en réalité à un serveur ou une passerelle hostile, l'attaquant peut demander un mot de passe, le stocker pour une utilisation ultérieure et simuler une erreur.

Cela étant dit, l'authentification de base reste utile pour personnaliser ou contrôler facilement l'accès aux documents dans un environnement convivial, ou lorsque la confidentialité est souhaitée sans être absolument nécessaire. Ainsi, l'authentification de base permet d'empêcher tout accès accidentel ou occasionnel par des utilisateurs curieux.*

Par exemple, au sein d'une entreprise, la gestion des produits peut protéger par mot de passe les plans de futurs produits afin de limiter leur diffusion prématurée. Une authentification de base rend l'accès à ces données suffisamment difficile pour des parties amies.† De même, vous pouvez protéger par mot de passe des photos personnelles ou des sites web privés qui ne sont pas top secrets ou ne contiennent pas d'informations précieuses, mais qui ne concernent pas non plus autrui.

L'authentification de base peut être sécurisée en la combinant à une transmission de données chiffrée (comme SSL) afin de dissimuler le nom d'utilisateur et le mot de passe aux personnes malveillantes. Il s'agit d'une technique courante.

Nous discutons du cryptage sécurisé au chapitre 14. Le chapitre suivant explique un protocole d'authentification HTTP plus sophistiqué, l'authentification Digest, qui possède des propriétés de sécurité plus fortes que l'authentification de base.

* Veillez à ce que le nom d'utilisateur/mot de passe de l'authentification de base ne soit pas le même que le mot de passe de vos systèmes plus sécurisés, sinon des utilisateurs malveillants peuvent les utiliser pour pénétrer dans vos comptes sécurisés !

† Bien que la sécurité ne soit pas optimale, les employés internes de l'entreprise sont généralement peu enclins à récupérer des mots de passe de manière malveillante. Cela dit, l'espionnage industriel existe bel et bien, et il existe des employés vengeurs et mécontents. Il est donc judicieux de placer toutes les données qui pourraient être très dangereuses si elles étaient acquises de manière malveillante sous un dispositif de sécurité renforcé.

Pour plus d'informations

Pour plus d'informations sur l'authentification de base et LDAP, consultez :

http://www.ietf.org/rfc/rfc2617.txt

RFC 2617, « Authentification HTTP : authentification d'accès de base et Digest ». http://www.ietf.org/rfc/rfc2616.txt

RFC 2616 « Protocole de transfert hypertexte — HTTP/1.1. »

Pour plus d'informations

Chapitre 13Ceci est le titre du livreCHAPITRE 13

Authentification Digest

L'authentification de base est pratique et flexible, mais totalement non sécurisée. Les noms d'utilisateur et les mots de passe sont envoyés en clair*, et aucune protection n'est mise en place contre toute falsification des messages. La seule façon d'utiliser l'authentification de base en toute sécurité est de l'utiliser conjointement avec SSL.

L'authentification Digest a été développée comme une alternative compatible et plus sécurisée à l'authentification de base. Ce chapitre est consacré à la théorie et à la pratique de l'authentification Digest. Bien que l'utilisation de l'authentification Digest soit encore limitée, ses concepts restent importants pour la mise en œuvre de transactions sécurisées.

Les améliorations de l'authentification Digest

L'authentification Digest est un protocole d'authentification HTTP alternatif qui tente de corriger les failles les plus importantes de l'authentification de base. Plus précisément, l'authentification Digest :

• N'envoie jamais de mots de passe secrets en clair sur le réseau

• Empêche les individus sans scrupules de capturer et de rejouer les poignées de main d'authentification

• Peut éventuellement protéger contre la falsification du contenu des messages

• Protège contre plusieurs autres formes courantes d’attaques

L'authentification Digest n'est pas le protocole le plus sécurisé possible.† De nombreux besoins en matière de transactions HTTP sécurisées ne peuvent être satisfaits par l'authentification Digest. Pour ces besoins, les protocoles TLS (Transport Layer Security) et HTTPS (Secure HTTP) sont plus adaptés.

* Les noms d'utilisateur et les mots de passe sont cryptés à l'aide d'un codage en base 64 simple et facilement déchiffrable. Ce procédé protège contre toute consultation accidentelle, mais n'offre aucune protection contre les personnes malveillantes.

† Par exemple, comparée aux mécanismes à clé publique, l'authentification Digest ne fournit pas de mécanisme d'authentification robuste. De plus, elle n'offre aucune protection de confidentialité au-delà de la protection du mot de passe : le reste de la requête et de la réponse sont accessibles aux indiscrets.

Cependant, l'authentification Digest est nettement plus puissante que l'authentification de base, qu'elle a été conçue pour remplacer. Elle est également plus puissante que de nombreux schémas populaires proposés pour d'autres services Internet, tels que CRAM-MD5, proposé pour une utilisation avec LDAP, POP et IMAP.

À ce jour, l'authentification Digest n'a pas été largement déployée. Cependant, en raison des risques de sécurité inhérents à l'authentification de base, les architectes HTTP recommandent, dans la RFC 2617, que « tout service actuellement utilisé utilisant l'authentification de base devrait passer à Digest dès que possible »*. Le succès de cette norme reste incertain.

Utiliser des résumés pour garder les mots de passe secrets

Le principe de l'authentification par condensé est de « ne jamais envoyer le mot de passe sur le réseau ». Au lieu d'envoyer le mot de passe, le client envoie une « empreinte digitale » ou « condensé », qui constitue un brouillage irréversible du mot de passe. Le client et le serveur connaissent tous deux le mot de passe secret, ce qui permet au serveur de vérifier que le condensé a fourni une correspondance correcte. Avec ce seul condensé, un pirate n'a aucun moyen simple de trouver le mot de passe d'origine, si ce n'est de passer en revue tous les mots de passe existants et de les essayer un par un !†

Voyons comment cela fonctionne (c'est une version simplifiée) :

• Dans la figure 13-1a, le client demande un document protégé.

• Dans la figure 13-1b, le serveur refuse de transmettre le document tant que le client n'a pas authentifié son identité en prouvant qu'il connaît le mot de passe. Le serveur lance un défi au client, lui demandant son nom d'utilisateur et une version condensée du mot de passe.

• Dans la figure 13-1c, le client prouve qu'il connaît le mot de passe en lui transmettant son résumé. Le serveur connaît les mots de passe de tous les utilisateurs‡ ; il peut donc vérifier que l'utilisateur connaît le mot de passe en comparant le résumé fourni par le client avec celui calculé en interne par le serveur. Un tiers aurait du mal à créer le bon résumé s'il ne connaissait pas le mot de passe.

• Dans la figure 13-1d, le serveur compare le condensé fourni par le client avec celui calculé en interne. Si les deux concordent, cela indique que le client connaît le mot de passe (ou a eu beaucoup de chance !). La fonction de condensé peut être configurée pour générer tellement

de chiffres que les conjectures sont impossibles. Lorsque le serveur vérifie la correspondance, le document est transmis au client, sans même envoyer le mot de passe sur le réseau.

* La pertinence de l'authentification Digest a fait l'objet d'un débat important, compte tenu de la popularité et de l'adoption généralisée du protocole HTTP chiffré SSL. L'avenir nous dira si l'authentification Digest atteindra la masse critique requise.

† Il existe des techniques, comme les attaques par dictionnaire, qui consistent à tester d'abord des mots de passe courants. Ces techniques de cryptanalyse peuvent considérablement faciliter le décryptage des mots de passe.

‡ En fait, le serveur n’a réellement besoin de connaître que les résumés des mots de passe.

Les améliorations de l'authentification Digest

Figure 13-1. Utilisation de résumés pour l'authentification masquée par mot de passe

Nous discuterons plus en détail des en-têtes particuliers utilisés dans l'authentification Digest dans le tableau 13-8.

Résumés à sens unique

Un condensé est une « condensation d'un ensemble d'informations ».* Les condensés agissent comme des fonctions unidirectionnelles, convertissant généralement un nombre infini de valeurs d'entrée possibles en une plage finie de condensations.† Une fonction de condensé populaire, MD5,‡ convertit n'importe quelle séquence arbitraire d'octets, de n'importe quelle longueur, en un condensé de 128 bits.

128 bits = 2128, soit environ 1 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 condensations distinctes possibles.

* Dictionnaire Merriam-Webster, 1998.

† En théorie, comme nous convertissons un nombre infini de valeurs d'entrée en un nombre fini de valeurs de sortie, il est possible que deux entrées distinctes correspondent au même condensé. C'est ce qu'on appelle une collision. En pratique, le nombre de sorties potentielles est si important que le risque de collision est infime et, pour la correspondance des mots de passe, négligeable.

‡ MD5 signifie « Message Digest #5 », un algorithme de synthèse de messages. L'algorithme de hachage sécurisé (SHA) est une autre fonction de synthèse de messages populaire.

L'important avec ces résumés, c'est que si vous ne connaissez pas le mot de passe secret, vous aurez beaucoup de mal à deviner le résumé correct à envoyer au serveur. De même, si vous possédez le résumé, vous aurez beaucoup de mal à déterminer laquelle des innombrables valeurs d'entrée l'a généré.

Les 128 bits de la sortie MD5 sont souvent écrits sous forme de 32 caractères hexadécimaux, chaque caractère représentant 4 bits. Le tableau 13-1 présente quelques exemples de condensés MD5 d'entrées d'échantillon. Notez comment MD5 prend des entrées arbitraires et produit un condensé de longueur fixe.

Tableau 13-1. Exemples de résumés MD5

Entrée du condensé MD5

«Salut» C1A5298F939E87E8F962A5EDFC206918

« bri:Aïe ! » BEAAA0E34EBDB072F8627C038AB211F8

« 3.1415926535897 » 475B977E19ECEE70835BC6DF46F4F6DE

« http://www.http-guide.com/index.htm » C617C0C7D1D05F66F595E22A4B0EAAA5

« Nous tenons pour évidentes ces vérités : tous les hommes sont créés égaux, ils sont dotés par leur Créateur de certains droits inaliénables, parmi lesquels la vie, la liberté et la recherche du bonheur. Pour garantir ces droits, des gouvernements sont institués parmi les hommes, tirant leurs justes pouvoirs du consentement des gouvernés. Chaque fois qu'une forme de gouvernement devient destructrice de ces fins, le peuple a le droit de la modifier ou de l'abolir, et d'instituer un nouveau gouvernement, en posant ses fondements sur des principes et en organisant ses pouvoirs selon la forme qui lui semblera la plus susceptible d'assurer sa sécurité et son bonheur. » 66C4EF58DA7CB956BD04233FBB64E0A4

Les fonctions de résumé sont parfois appelées sommes de contrôle cryptographiques, fonctions de hachage unidirectionnelles ou fonctions d'empreintes digitales.

Utiliser des nonces pour empêcher les rediffusions

Les résumés unidirectionnels nous évitent d'avoir à envoyer les mots de passe en clair. Nous pouvons simplement envoyer un résumé du mot de passe, et être assurés qu'aucun tiers malveillant ne pourra facilement déchiffrer le mot de passe original à partir du résumé.

Malheureusement, les mots de passe obscurcis ne suffisent pas à nous protéger du danger, car un pirate peut capturer le résumé et le rejouer indéfiniment sur le serveur, même s'il ne connaît pas le mot de passe. Le résumé est tout aussi efficace que le mot de passe.

Pour empêcher de telles attaques par relecture, le serveur peut transmettre au client un jeton spécial appelé nonce*, qui change fréquemment (peut-être toutes les millisecondes, ou pour chaque

* Le mot nonce signifie « l'occasion présente » ou « le moment présent ». En matière de sécurité informatique, le nonce capture un instant précis et l'intègre dans les calculs de sécurité.

Les améliorations de l'authentification Digest

authentification). Le client ajoute ce jeton nonce au mot de passe avant de calculer le condensé.

Mélanger le nonce et le mot de passe entraîne une modification du condensé à chaque changement de nonce. Cela empêche les attaques par rejeu, car le condensé de mot de passe enregistré n'est valide que pour une valeur de nonce particulière, et sans le mot de passe secret, l'attaquant ne peut pas calculer le condensé correct.

L'authentification Digest nécessite l'utilisation de nonces, car une faiblesse de relecture triviale rendrait l'authentification Digest sans nonces aussi faible que l'authentification de base. Les nonces sont transmis du serveur au client lors du défi WWW-Authenticate.

La poignée de main d'authentification Digest

Le protocole d'authentification HTTP Digest est une version améliorée de l'authentification qui utilise des en-têtes similaires à ceux de l'authentification de base. De nouvelles options ont été ajoutées aux en-têtes traditionnels, ainsi qu'un nouvel en-tête facultatif, Authorization-Info.

La poignée de main simplifiée en trois phases de l’authentification digest est illustrée dans la Figure 13-2.

Figure 13-2. Poignée de main d'authentification Digest

Voici ce qui se passe dans la figure 13-2 :

• À l'étape 1, le serveur calcule une valeur de nonce. À l'étape 2, il l'envoie au client dans un message de défi WWW-Authenticate, accompagné d'une liste d'algorithmes pris en charge.

• À l'étape 3, le client sélectionne un algorithme et calcule le résumé du mot de passe secret et des autres données. À l'étape 4, il renvoie le résumé au serveur dans un message d'autorisation. Si le client souhaite authentifier le serveur, il peut lui envoyer un nonce client.

À l'étape 5, le serveur reçoit le condensé, l'algorithme choisi et les données de support, et calcule le même condensé que le client. Le serveur compare ensuite le condensé généré localement à celui

transmis par le réseau et valide leur correspondance. Si le client a symétriquement interrogé le serveur avec un nonce client, un condensé client est créé. De plus, le nonce suivant peut être précalculé et transmis au client à l'avance, afin que celui-ci puisse émettre le bon condensé la prochaine fois.

Nombre de ces informations sont facultatives et possèdent des valeurs par défaut. Pour clarifier les choses, la figure 13-3 compare les messages envoyés pour l'authentification de base (figures 13-3a à d) avec un exemple simple d'authentification par résumé (figures 13-3e à h).

Examinons maintenant de plus près le fonctionnement interne de l'authentification Digest.

Calculs de synthèse

Le cœur de l'authentification par condensé est le condensé unidirectionnel d'un mélange d'informations publiques, d'informations secrètes et d'une valeur de nonce limitée dans le temps. Voyons maintenant comment les condensés sont calculés. Les calculs de condensé sont généralement simples.* Un exemple de code source est fourni en annexe F.

Données d'entrée de l'algorithme de digestion

Les résumés sont calculés à partir de trois éléments :

• Une paire de fonctions composée d'une fonction de hachage unidirectionnelle H(d) et d'un condensé

KD(s,d), où s représente secret et d représente données

• Un bloc de données contenant des informations de sécurité, y compris le mot de passe secret, appelé A1

• Un bloc de données contenant des attributs non secrets du message de demande, appelé A2. Les deux éléments de données, A1 et A2, sont traités par H et KD pour produire un condensé.

Les algorithmes H(d) et KD(s,d)

L'authentification Digest prend en charge plusieurs algorithmes Digest. Les deux algorithmes suggérés dans la RFC 2617 sont MD5 et MD5-sess (où « sess » signifie session). L'algorithme par défaut est MD5 si aucun autre algorithme n'est spécifié.

* Cependant, les modes de compatibilité optionnels de la RFC 2617 et le manque de documentation dans les spécifications compliquent légèrement la tâche des débutants. Nous allons essayer de vous aider…

Figure 13-3. Syntaxe d'authentification de base et syntaxe d'authentification digest

Si MD5 ou MD5-sess est utilisé, la fonction H calcule le MD5 des données, et la fonction KD Digest calcule le MD5 des données secrètes et non secrètes jointes par deux points. Autrement dit :

H(<données>) = MD5(<données>)

KD(<secret>,<données>) = H(concaténer(<secret>:<données>))

Les données relatives à la sécurité (A1)

Le bloc de données appelé A1 est un produit d'informations secrètes et de protection, telles que le nom d'utilisateur, le mot de passe, le domaine de protection et les nonces. A1 concerne uniquement les informations de sécurité, et non le message sous-jacent lui-même. A1 est utilisé avec H, KD et A2 pour calculer les résumés.

La RFC 2617 définit deux manières de calculer A1, selon l'algorithme choisi :

MD5

Des hachages unidirectionnels sont exécutés pour chaque demande ; A1 est le triple joint par deux points du nom d'utilisateur, du domaine et du mot de passe secret.

Session MD5

La fonction de hachage n'est exécutée qu'une seule fois, lors de la première négociation WWW-Authenticate ; le hachage gourmand en CPU du nom d'utilisateur, du domaine et du mot de passe secret est effectué une fois et ajouté aux valeurs actuelles du nonce et du nonce client (cnonce).

Les définitions de A1 sont présentées dans le tableau 13-2.

Tableau 13-2. Définitions de A1 par algorithme

Algorithme A1

MD5 A1 = <utilisateur>:<domaine>:<mot de passe>

MD5-sess A1 = MD5(<utilisateur>:<domaine>:<mot de passe>):<nonce>:<cnonce>

Les données relatives aux messages (A2)

Le bloc de données appelé A2 représente des informations sur le message lui-même, telles que l'URL, la méthode de requête et le corps de l'entité du message. A2 est utilisé pour protéger contre la falsification de la méthode, de la ressource ou du message. A2 est utilisé avec H, KD et A1 pour calculer les résumés.

La RFC 2617 définit deux schémas pour A2, en fonction de la qualité de protection (qop) choisie :

• Le premier schéma implique uniquement la méthode de requête HTTP et l'URL. Il est utilisé lorsque qop=« auth », ce qui est le cas par défaut.

• Le deuxième schéma ajoute le corps de l'entité du message pour assurer un certain degré de vérification de l'intégrité du message. Il est utilisé lorsque qop=« auth-int ».

Les définitions de A2 sont présentées dans le tableau 13-3.

Tableau 13-3. Définitions de A2 par algorithme (résumés de requête)

QOP A2

<méthode-de-requête> indéfinie : <valeur-de-directive-uri>

auth <méthode-de-requête>:<valeur-de-directive-uri>

auth-int <méthode-de-requête>:<valeur-de-directive-uri>:H(<corps-d'entité-de-requête>)

La méthode request-method est la méthode de requête HTTP. La valeur uri-directive-value est l'URI de la requête. Il peut s'agir de « * », d'une « absoluteURL » ou d'un « abs_path », mais elle doit correspondre à l'URI de la requête. En particulier, il doit s'agir d'une URL absolue si l'URI de la requête est une absoluteURL.

Algorithme de résumé global

La RFC 2617 définit deux manières de calculer les condensés, étant donné H, KD, A1 et A2 :

• La première méthode est destinée à être compatible avec l'ancienne spécification RFC2069, utilisée lorsque l'option qop est manquante. Elle calcule le condensé à partir du hachage des informations secrètes et des données du message nonced.

• La deuxième approche est l'approche moderne et privilégiée : elle inclut la prise en charge du décompte nonce et de l'authentification symétrique. Cette approche est utilisée lorsque la qualité de l'opération est

« auth » ou « auth-int ». Ce paramètre ajoute les données de nombre de nonces, de QOP et de cnonce au condensé.

Les définitions de la fonction de résumé résultante sont présentées dans le tableau 13-4. Notez que les résumés résultants utilisent H, KD, A1 et A2.

Tableau 13-4. Anciens et nouveaux algorithmes de synthèse

Notes sur l'algorithme qop Digest

KD(H(A1), <nonce>:H(A2)) non défini

auth ou auth-int KD(H(A1), <nonce>:<nc>:<cnonce>:<qop>:H(A2)) Préféré

Il est facile de se perdre dans toutes les couches de l'encapsulation dérivationnelle. C'est l'une des raisons pour lesquelles certains lecteurs ont des difficultés avec la RFC 2617. Pour simplifier les choses, le tableau 13-5 développe les définitions de H et KD et conserve les résumés en termes de A1 et A2.

Tableau 13-5. Aide-mémoire de l'algorithme de synthèse déplié

Algorithme qop Algorithme déplié

indéfini <indéfini> MD5(MD5(A1):<nonce>:MD5(A2))

MD5 MD5-sess Tableau 13-5. Aide-mémoire de l'algorithme de synthèse déplié (suite)

Algorithme qop Algorithme déplié

auth <indéfini>

MD5

Session MD5 MD5(MD5(A1):<nonce>:<nc>:<cnonce>:<qop>:MD5(A2))

auth-int <indéfini>

MD5

Session MD5 MD5(MD5(A1):<nonce>:<nc>:<cnonce>:<qop>:MD5(A2))

Session d'authentification Digest

La réponse du client à un défi WWW-Authenticate pour un espace de protection démarre une session d'authentification avec cet espace de protection (le domaine combiné à la racine canonique du serveur auquel on accède définit un « espace de protection »).

La session d'authentification dure jusqu'à ce que le client reçoive une nouvelle demande WWW-Authenticate d'un serveur de l'espace de protection. Le client doit mémoriser le nom d'utilisateur, le mot de passe, le nonce, le nombre de nonces et les valeurs opaques associés à une session d'authentification afin de les utiliser pour construire l'en-tête d'autorisation des futures requêtes dans cet espace de protection.

À l'expiration du nonce, le serveur peut choisir d'accepter les anciennes informations de l'en-tête d'autorisation, même si la valeur du nonce incluse n'est pas récente. Le serveur peut également renvoyer une réponse 401 avec une nouvelle valeur de nonce, obligeant le client à réessayer la requête. En spécifiant « stale=true » dans cette réponse, le serveur indique au client de réessayer avec le nouveau nonce sans demander de nouveaux nom d'utilisateur et mot de passe.

Autorisation préemptive

En authentification normale, chaque requête nécessite un cycle de requête/défi avant que la transaction puisse être finalisée. Ceci est illustré à la figure 13-4a.

Ce cycle de requête/défi peut être éliminé si le client connaît à l'avance le prochain nonce, ce qui lui permet de générer l'en-tête d'autorisation correct avant que le serveur ne le demande. Si le client peut calculer l'en-tête d'autorisation avant sa demande, il peut l'envoyer de manière préventive au serveur, sans passer par une requête/défi au préalable. L'impact sur les performances est illustré à la figure 13-4b.

L'autorisation préventive est triviale (et courante) pour l'authentification de base. Les navigateurs gèrent généralement des bases de données côté client contenant les noms d'utilisateur et les mots de passe. Une fois qu'un utilisateur s'est authentifié sur un site, le navigateur envoie généralement l'en-tête d'autorisation correct pour les requêtes ultérieures à cette URL (voir chapitre 12).

Figure 13-4. L'autorisation préventive réduit le nombre de messages

L'autorisation préventive est un peu plus complexe pour l'authentification Digest, en raison de la technologie des nonces destinée à déjouer les attaques par rejeu. Le serveur générant des nonces arbitraires, le client n'a pas toujours la possibilité de déterminer l'en-tête d'autorisation à envoyer avant de recevoir une demande.

L'authentification Digest offre plusieurs moyens d'autorisation préventive tout en conservant de nombreuses fonctionnalités de sécurité. Voici trois façons pour un client d'obtenir le nonce correct sans attendre un nouveau défi WWW-Authenticate :

• Le serveur pré-envoie le nonce suivant dans l'en-tête de réussite Authentication-Info.

• Le serveur permet de réutiliser le même nonce pendant une petite fenêtre de temps.

• Le client et le serveur utilisent tous deux un algorithme de génération de nonce synchronisé et prévisible.

Prégénération du nonce suivant

La valeur de nonce suivante peut être fournie à l'avance au client par l'en-tête de réussite AuthenticationInfo. Cet en-tête est envoyé avec la réponse 200 OK d'une précédente authentification réussie.

Informations d'authentification : nextnonce="<valeur-nonce>"

Étant donné le nonce suivant, le client peut émettre de manière préventive un en-tête d'autorisation.

Bien que cette autorisation préventive évite un cycle requête/défi (accélérant ainsi la transaction), elle empêche également l'acheminement de plusieurs requêtes vers le même serveur, car la valeur nonce suivante doit être reçue avant l'émission de la requête suivante. Le pipelining étant considéré comme une technologie fondamentale pour éviter la latence, la baisse de performance peut être importante.

Réutilisation limitée des nonces

Au lieu de prégénérer une séquence de nonces, une autre approche consiste à autoriser une réutilisation limitée des nonces. Par exemple, un serveur peut autoriser la réutilisation d'un nonce 5 fois, soit pendant 10 secondes.

Dans ce cas, le client peut émettre librement des requêtes avec l'en-tête Authorization et les canaliser, car le nonce est connu à l'avance. À l'expiration du nonce, le serveur doit envoyer au client une requête 401 Unauthorized, avec la directive WWW-Authenticate: stale=true définie :

Authentification WWW : Digest realm="<valeur-du-domaine>" nonce="<valeur-du-nonce>" stale=true

La réutilisation des nonces réduit la sécurité, car elle facilite la réussite des attaques par rejeu par un attaquant. La durée de réutilisation des nonces étant contrôlable, de l'absence totale de réutilisation à une réutilisation potentiellement longue, des compromis peuvent être trouvés entre les fenêtres de vulnérabilité et les performances.

De plus, d'autres fonctionnalités peuvent être utilisées pour rendre les attaques par rejeu plus difficiles, notamment l'incrémentation des compteurs et les tests d'adresses IP. Cependant, si ces techniques rendent les attaques plus difficiles, elles n'éliminent pas la vulnérabilité.

Génération de nonce synchronisée

Il est possible d'utiliser des algorithmes de génération de nonces synchronisés dans le temps, où le client et le serveur peuvent générer une séquence de nonces identiques, sur la base d'une clé secrète partagée, qu'un tiers ne peut pas facilement prédire (comme les cartes d'identité sécurisées).

Ces algorithmes dépassent le cadre de la spécification d’authentification Digest.

Sélection de nonce

Le contenu du nonce est opaque et dépend de l'implémentation. Cependant, la qualité des performances, la sécurité et la commodité dépendent d'un choix judicieux.

La RFC 2617 suggère cette formulation hypothétique de nonce :

BASE64(horodatage H(horodatage ":" ETag ":" clé-privée)) où horodatage est une heure générée par le serveur ou une autre valeur non répétitive, ETag est la valeur de l'en-tête HTTP ETag associé à l'entité demandée et clé-privée est une donnée connue uniquement du serveur.

Avec un nonce de cette forme, le serveur recalcule la partie hachage après réception de l'en-tête d'authentification du client et rejette la requête si elle ne correspond pas au nonce de cet en-tête ou si la valeur d'horodatage n'est pas suffisamment récente. De cette façon, le serveur peut limiter la durée de validité du nonce.

L'inclusion de l'ETag empêche toute demande de relecture pour une version mise à jour de la ressource. (Notez que l'inclusion de l'adresse IP du client dans le nonce semble offrir au serveur la possibilité de limiter la réutilisation du nonce au client qui l'a initialement reçu. Cependant, cela perturberait les fermes de proxys, dans lesquelles les requêtes d'un même utilisateur transitent souvent par différents proxys. De plus, l'usurpation d'adresse IP n'est pas si compliquée.)

Une implémentation peut choisir de ne pas accepter un nonce ou un condensé déjà utilisé afin de se protéger contre les attaques par rejeu. Elle peut également choisir d'utiliser des nonces ou des condensés à usage unique pour les requêtes POST ou PUT, et des horodatages pour les requêtes GET.

Reportez-vous à « Considérations de sécurité » pour les considérations de sécurité pratiques qui affectent la sélection du nonce.

Authentification symétrique

La RFC 2617 étend l'authentification par condensé pour permettre au client d'authentifier le serveur. Pour ce faire, elle fournit une valeur de nonce client, à partir de laquelle le serveur génère un condensé de réponse correct, basé sur une connaissance correcte des informations secrètes partagées. Le serveur renvoie ensuite ce condensé au client dans l'en-tête Authorization-Info.

Cette authentification symétrique est standard depuis la RFC 2617. Elle est facultative pour des raisons de rétrocompatibilité avec l'ancienne norme RFC 2069. Cependant, comme elle apporte d'importantes améliorations de sécurité, il est fortement recommandé à tous les clients et serveurs modernes d'implémenter toutes les fonctionnalités de la RFC 2617. En particulier, l'authentification symétrique doit être effectuée dès qu'une directive qop est présente et non en son absence.

Le résumé de réponse est calculé comme le résumé de requête, à la différence que les informations du corps du message (A2) sont différentes, car la réponse ne contient aucune méthode et que les données d'entité du message sont différentes. Les méthodes de calcul

de A2 pour les résumés de requête et de réponse sont comparées dans les tableaux 13-6 et 13-7.

Tableau 13-6. Définitions de A2 par algorithme (résumés de requête)

QOP A2

<méthode-de-requête> indéfinie : <valeur-de-directive-uri>

auth <méthode-de-requête>:<valeur-de-directive-uri>

auth-int <méthode-de-requête>:<valeur-de-directive-uri>:H(<corps-d'entité-de-requête>)

Tableau 13-7. Définitions de A2 par algorithme (résumés de réponses)

QOP A2

indéfini :<valeur-de-directive-uri>

auth :<valeur-de-directive-uri>

auth-int :<valeur-de-directive-uri>:H(<corps-d'entité-de-réponse>)

Les valeurs cnonce et nc doivent correspondre à celles de la requête client à laquelle ce message répond. Les directives response auth, cnonce et nonce count doivent être présentes si qop="auth" ou qop="auth-int" est spécifié.

Améliorations de la qualité de la protection

Le champ qop peut être présent dans les trois en-têtes de résumé : WWW-Authenticate,

Autorisation et informations d'authentification.

Le champ QOP permet aux clients et aux serveurs de négocier différents types et niveaux de protection. Par exemple, certaines transactions peuvent vouloir vérifier l'intégrité du corps des messages, même si cela ralentit considérablement la transmission.

Le serveur exporte d'abord une liste d'options QOP séparées par des virgules dans l'en-tête WWW-Authenticate. Le client sélectionne ensuite l'option qu'il prend en charge et qui répond à ses besoins, puis la renvoie au serveur dans son champ QOP d'autorisation.

L'utilisation de l'option qop est facultative, mais uniquement pour des raisons de compatibilité avec l'ancienne spécification RFC 2069. L'option qop devrait être prise en charge par tous les digests modernes.

implémentations.

La RFC 2617 définit deux valeurs initiales de qualité de protection : « auth », qui indique l'authentification, et « auth-int », qui indique

l'authentification avec protection de l'intégrité du message. D'autres options de qualité de protection sont attendues ultérieurement.

Protection de l'intégrité des messages

Si la protection d'intégrité est appliquée (qop="auth-int"), H (le corps de l'entité) correspond au hachage du corps de l'entité, et non du corps du message. Il est calculé avant l'application de tout codage de transfert par l'expéditeur et après sa suppression par le destinataire. Notez que cela inclut les limites multiparties et les en-têtes intégrés dans chaque partie de tout type de contenu multipartie.

Améliorations de la qualité de la protection

En-têtes d'authentification Digest

Les protocoles d'authentification de base et digest contiennent tous deux un défi d'autorisation, porté par l'en-tête WWW-Authenticate, et une réponse d'autorisation, portée par l'en-tête Authorization. L'authentification digest ajoute un en-tête Authorization-Info facultatif, envoyé après une authentification réussie, pour conclure une négociation en trois phases et transmettre le prochain nonce à utiliser. Les en-têtes d'authentification de base et digest sont présentés dans le tableau 13-8.

Tableau 13-8. En-têtes d'authentification HTTP

Phase Basic Digest

Défi WWW-Authenticate : Basic realm="<realm-value>" WWW-Authenticate : Digest realm="<realm-value>" nonce="<nonce-value>"

[domain="<liste-d'URI>"]

[opaque="<opaque-token-value>"]

[stale=<vrai-ou-faux>]

[algorithme=<algorithme-digest>]

[qop="<liste-de-valeurs-qop>"]

[<extension-directive>]

Autorisation de réponse : de base

<base64(user:pass)> Autorisation : Digest username="<username>" realm="<realm-value>" nonce="<nonce-value>" uri=<request-uri> response="<32-hex-digit-digest>" [algorithm=<digest-algorithm>]

[opaque="<opaque-token-value>"]

[cnonce="<valeur-nonce>"]

[qop=<valeur-qop>]

[nc=<8-hex-digit-nonce-count>]

[<extension-directive>]

Info n/a Authentification-Info :

nextnonce="<valeur-nonce>"

[qop="<liste-de-valeurs-qop>"]

[rspauth="<hex-digest>"]

[cnonce="<valeur-nonce>"]

[nc=<8-hex-digit-nonce-count>]

Les en-têtes d'authentification Digest sont un peu plus complexes. Ils sont décrits en détail dans l'annexe F.

Considérations pratiques

Plusieurs éléments doivent être pris en compte lors de l'utilisation de l'authentification Digest. Cette section aborde certains de ces points.

De multiples défis

Un serveur peut émettre plusieurs défis pour une ressource. Par exemple, s'il ne connaît pas les capacités d'un client, il peut lui proposer des défis d'authentification de base et des défis d'authentification digest. Face à ces défis, le client doit choisir le mécanisme d'authentification le plus puissant qu'il prend en charge.

Les agents utilisateurs doivent être particulièrement vigilants lors de l'analyse de la valeur du champ d'en-tête WWW-Authenticate ou ProxyAuthenticate si celui-ci contient plusieurs défis ou si plusieurs champs d'en-tête WWW-Authenticate sont fournis, car un défi peut lui-même contenir une liste de paramètres d'authentification séparés par des virgules. Notez que de nombreux navigateurs ne reconnaissent que l'authentification de base et exigent qu'elle soit le premier mécanisme d'authentification présenté.

La fourniture d'un large éventail d'options d'authentification soulève des problèmes évidents de sécurité liés au « maillon faible ». Les serveurs ne devraient inclure l'authentification de base que si elle est minimalement acceptable, et les administrateurs devraient avertir les utilisateurs des dangers liés au partage du même mot de passe entre plusieurs systèmes lorsque différents niveaux de sécurité sont utilisés.

Gestion des erreurs

Dans l'authentification Digest, si une directive ou sa valeur est incorrecte, ou si une directive requise est manquante, la réponse appropriée est 400 Bad Request.

Si le résumé d'une requête ne correspond pas, un échec de connexion doit être enregistré. Des échecs répétés d'un client peuvent indiquer qu'un attaquant tente de deviner les mots de passe.

Le serveur d'authentification doit s'assurer que la ressource désignée par la directive « uri » est identique à celle spécifiée dans la ligne de requête ; si elles sont différentes, le serveur doit renvoyer une erreur 400 « Requête incorrecte ». (Comme cela peut être le symptôme d'une attaque, les concepteurs de serveurs peuvent envisager de consigner ces erreurs.) La duplication des informations de l'URL de la requête dans ce champ permet d'éviter qu'un proxy intermédiaire ne modifie la ligne de requête du client. Cette requête modifiée (mais vraisemblablement sémantiquement équivalente) ne produirait pas le même condensé que celui calculé par le client.

Espaces de protection

La valeur du domaine, combinée à l'URL racine canonique du serveur auquel on accède, définit l'espace de protection.

Les domaines permettent de partitionner les ressources protégées d'un serveur en un ensemble d'espaces de protection, chacun doté de son propre schéma d'authentification et/ou de sa propre base de données d'autorisation. La valeur du domaine est une chaîne, généralement attribuée par le serveur d'origine, qui peut contenir des informations spécifiques au schéma d'authentification. Notez que plusieurs problèmes peuvent survenir avec un même schéma d'autorisation, mais avec des domaines différents.

Considérations pratiques

L'espace de protection détermine le domaine sur lequel les informations d'identification peuvent être appliquées automatiquement. Si une requête antérieure a été autorisée, les mêmes informations d'identification peuvent être réutilisées pour toutes les autres requêtes au sein de cet espace de protection pendant une durée déterminée par le schéma d'authentification, les paramètres et/ou les préférences de l'utilisateur. Sauf indication contraire du schéma d'authentification, un espace de protection unique ne peut pas s'étendre au-delà de la portée de son serveur.

Le calcul spécifique de l'espace de protection dépend du mécanisme d'authentification :

• Dans l'authentification de base, les clients supposent que tous les chemins situés au niveau ou en dessous de l'URI de la requête se trouvent dans le même espace de protection que le défi actuel. Un client peut autoriser de manière préventive les ressources dans cet espace sans attendre un autre défi du serveur.

• Dans l'authentification Digest, le champ WWW-Authenticate: domain du défi définit plus précisément l'espace de protection. Ce champ est une liste d'URI entre guillemets et séparés par des espaces. Tous les URI de la liste de domaines, ainsi que tous ceux logiquement sous ces préfixes, sont supposés appartenir au même espace de protection. Si le

champ domain est manquant ou vide, tous les URI du serveur du défi se trouvent dans l'espace de protection.

Réécriture des URI

Les proxys peuvent réécrire les URI de manière à modifier leur syntaxe, mais pas la ressource décrite. Par exemple :

• Les noms d'hôtes peuvent être normalisés ou remplacés par des adresses IP.

• Les caractères intégrés peuvent être remplacés par des formes d'échappement « % ».

• Des attributs supplémentaires d'un type qui n'affecte pas la ressource extraite du serveur d'origine particulier peuvent être ajoutés ou insérés dans l'URI.

Étant donné que les URI peuvent être modifiés par les proxys et que l'authentification Digest vérifie l'intégrité de la valeur de l'URI, l'authentification Digest sera interrompue si l'une de ces modifications est effectuée. Voir « Données relatives aux messages (A2) » pour plus d'informations.

Caches

Lorsqu'un cache partagé reçoit une requête contenant un en-tête d'autorisation et une réponse du relais de cette requête, il ne doit pas renvoyer cette réponse en réponse à une autre requête, à moins que l'une des deux directives Cache-Control ne soit présente dans la réponse :

• Si la réponse d'origine incluait la directive Cache-Control « must-revalidate », le cache peut utiliser l'entité de cette réponse pour répondre à une requête ultérieure. Cependant, il doit d'abord la revalider auprès du serveur d'origine, en utilisant les en-têtes de la nouvelle requête, afin que le serveur d'origine puisse authentifier la nouvelle requête.

• Si la réponse d'origine incluait la directive Cache-Control « public », l'entité de réponse peut être renvoyée en réponse à toute demande ultérieure.

Considérations de sécurité

La RFC 2617 résume admirablement certains des risques de sécurité inhérents aux schémas d'authentification HTTP. Cette section décrit certains de ces risques.

Falsification de l'en-tête

Pour garantir une protection infaillible contre la falsification des en-têtes, il est nécessaire d'utiliser un chiffrement de bout en bout ou une signature numérique des en-têtes, de préférence une combinaison des

deux ! L'authentification Digest vise à fournir un système d'authentification inviolable, mais n'étend pas nécessairement cette protection aux données. Les seuls en-têtes offrant un certain niveau de protection sont WWW-Authenticate et Authorization.

Attaques par relecture

Dans le contexte actuel, une attaque par rejeu se produit lorsqu'une personne utilise un ensemble d'identifiants d'authentification espionnés d'une transaction donnée pour une autre transaction. Bien que ce problème concerne les requêtes GET, il est essentiel de disposer d'une méthode infaillible pour éviter les attaques par rejeu pour les requêtes POST et PUT. La possibilité de rejouer avec succès des identifiants précédemment utilisés lors du transport de données de formulaire pourrait engendrer des problèmes de sécurité.

Ainsi, pour qu'un serveur accepte les identifiants « rejoués », les valeurs de nonce doivent être répétées. Une solution pour atténuer ce problème consiste à demander au serveur de générer un nonce contenant un résumé de l'adresse IP du client, un horodatage, l'ETag de la ressource et une clé de serveur privée (comme recommandé précédemment). Dans un tel scénario, la combinaison d'une adresse IP et d'un délai d'expiration court peut constituer un obstacle majeur pour l'attaquant.

Cependant, cette solution présente un inconvénient majeur. Comme nous l'avons vu précédemment, l'utilisation de l'adresse IP du client pour créer un nonce interrompt la transmission via les fermes de proxys, où les requêtes d'un même utilisateur peuvent transiter par différents proxys. De plus, l'usurpation d'adresse IP n'est pas trop complexe.

Une façon d'éviter complètement les attaques par rejeu consiste à utiliser une valeur de nonce unique pour chaque transaction. Dans cette implémentation, pour chaque transaction, le serveur émet une valeur de nonce unique accompagnée d'une valeur de délai d'expiration. Cette valeur de nonce émise n'est valide que pour la transaction concernée et uniquement pendant la durée du délai d'expiration. Cette comptabilisation peut augmenter la charge des serveurs ; toutefois, cette augmentation devrait être minime.

Mécanismes d'authentification multiples

Lorsqu'un serveur prend en charge plusieurs schémas d'authentification (tels que Basic et Digest), il propose généralement ce choix dans les en-têtes WWW-Authenticate. Le client étant

Considérations de sécurité

il n'est pas nécessaire d'opter pour le mécanisme d'authentification le plus fort, la force de l'authentification résultante n'est aussi bonne que celle du plus faible des schémas d'authentification.

La solution la plus évidente pour éviter ce problème est de demander aux clients de toujours choisir le schéma d'authentification le plus fort disponible. Si cela n'est pas réalisable (la plupart d'entre nous utilisent des clients commerciaux), la seule autre option consiste à utiliser un serveur proxy pour ne conserver que le schéma d'authentification le plus fort. Cependant, une telle approche n'est envisageable que dans un domaine où tous les clients sont reconnus pour prendre en charge le schéma d'authentification choisi, par exemple un réseau d'entreprise.

Attaques par dictionnaire

Les attaques par dictionnaire sont des attaques classiques de devinette de mots de passe. Un utilisateur malveillant peut intercepter une transaction et utiliser un programme standard de devinette de mots de passe pour les paires nonce/réponse. Si les utilisateurs utilisent des mots de passe relativement simples et les serveurs des nonces simplistes, il est tout à fait possible de trouver une correspondance. En l'absence de politique de vieillissement des mots de passe, avec le temps et le coût unique de leur décryptage, il est facile de collecter suffisamment de mots de passe pour causer de réels dommages.

Il n’existe pas vraiment de bonne façon de résoudre ce problème, si ce n’est en utilisant des mots de passe relativement complexes et difficiles à déchiffrer et une bonne politique de vieillissement des mots de passe.

Proxys hostiles et attaques de l'homme du milieu

Aujourd'hui, une grande partie du trafic Internet passe par un proxy à un moment ou à un autre. Avec l'avènement des techniques de redirection et des proxys d'interception, un utilisateur peut ne même pas se rendre compte que sa requête transite par un proxy. Si l'un de ces proxys est hostile ou compromis, le client peut être vulnérable à une attaque de l'homme du milieu.

Une telle attaque pourrait prendre la forme d’une écoute clandestine ou d’une modification des schémas d’authentification disponibles en supprimant tous les choix proposés et en les remplaçant par le schéma d’authentification le plus faible (comme l’authentification de base).

L'une des façons de compromettre un proxy de confiance est d'utiliser ses interfaces d'extension. Les proxys offrent parfois des interfaces de programmation sophistiquées, et il est possible d'y développer une extension (un plug-in) pour intercepter et modifier le trafic. Cependant, la sécurité des centres de données et celle offerte par les proxys eux-mêmes rendent le risque d'attaques de l'homme du milieu via des plug-ins malveillants assez faible.

Il n'existe aucune solution miracle à ce problème. Parmi les solutions possibles, on peut citer l'affichage d'indices visuels sur la force de

l'authentification par les clients, leur configuration pour qu'ils utilisent toujours l'authentification la plus forte possible, etc. Cependant, même avec ce schéma d'authentification, les clients restent vulnérables aux écoutes clandestines. Le seul moyen infaillible de se protéger contre ces attaques est d'utiliser SSL.

Attaques en texte clair choisies

Les clients utilisant l'authentification Digest utilisent un nonce fourni par le serveur pour générer la réponse. Cependant, si un proxy compromis ou malveillant intercepte le trafic (ou un serveur d'origine malveillant), il peut facilement fournir un nonce pour le calcul de la réponse par le client. L'utilisation de la clé connue pour le calcul de la réponse peut faciliter la cryptanalyse de celle-ci. C'est ce qu'on appelle une attaque par texte clair choisi. Il existe plusieurs variantes d'attaques par texte clair choisi :

Attaques par dictionnaire précalculé

Il s'agit d'une combinaison d'attaque par dictionnaire et d'attaque par texte clair choisi. Le serveur attaquant génère d'abord un ensemble de réponses, utilisant un nonce prédéterminé et des variantes de mots de passe courantes, et crée un dictionnaire. Une fois ce dictionnaire suffisamment volumineux disponible, le serveur/proxy attaquant peut finaliser l'interdiction du trafic et commencer à envoyer des nonces prédéterminés aux clients. Lorsqu'il reçoit une réponse d'un client, l'attaquant recherche des correspondances dans le dictionnaire généré. Si une correspondance est trouvée, l'attaquant dispose du mot de passe de cet utilisateur.

Attaques par force brute par lots

La différence avec une attaque par force brute par lots réside dans le calcul du mot de passe. Au lieu de chercher à faire correspondre un condensé précalculé, un ensemble de machines énumère tous les mots de passe possibles pour un espace donné. Plus les machines sont rapides, plus l'attaque par force brute devient viable.

En général, la menace posée par ces attaques est facile à contrer. Une solution consiste à configurer les clients pour qu'ils utilisent la directive facultative cnonce, afin que la réponse soit générée à la discrétion du client, et non le nonce fourni par le serveur (qui pourrait être compromis par l'attaquant). Ceci, combiné à des politiques imposant des mots de passe relativement forts et à un mécanisme efficace de vieillissement des mots de passe, peut atténuer complètement la menace des attaques par texte clair choisi.

Stockage des mots de passe

Le mécanisme d'authentification digest compare la réponse de l'utilisateur à ce qui est stocké en interne par le serveur, généralement

les noms d'utilisateur et les tuples H(A1), où H(A1) est dérivé du condensé du nom d'utilisateur, du domaine et du mot de passe.

Contrairement à un fichier de mot de passe traditionnel sur une machine Unix, si un fichier de mot de passe d'authentification Digest est compromis, tous les documents du domaine sont immédiatement disponibles pour l'attaquant ; aucune étape de décryptage n'est nécessaire.

Voici quelques moyens d'atténuer ce problème :

• Protégez le fichier de mots de passe comme s'il contenait des mots de passe en texte clair.

• Assurez-vous que le nom de domaine est unique parmi tous les domaines, afin qu'en cas de compromission d'un fichier de mots de passe, les dommages soient localisés à un domaine particulier. Un nom de domaine complet, incluant l'hôte et le domaine, devrait satisfaire à cette exigence.

Considérations de sécurité

Bien que l'authentification Digest offre une solution beaucoup plus robuste et sécurisée que l'authentification de base, elle n'offre toujours aucune protection pour la sécurité du contenu : une transaction véritablement sécurisée n'est possible que via SSL, que nous décrivons dans le chapitre suivant.

Pour plus d'informations

Pour plus d'informations sur l'authentification, voir :

http://www.ietf.org/rfc/rfc2617.txt

RFC 2617, « Authentification HTTP : authentification d'accès de base et Digest ».

Chapitre 14hCeci est le titre du livre CHAPITRE 14

HTTP sécurisé

Les trois chapitres précédents ont passé en revue les fonctionnalités de HTTP permettant d'identifier et d'authentifier les utilisateurs. Ces techniques fonctionnent bien dans une communauté amicale, mais elles ne sont pas assez puissantes pour protéger les transactions importantes d'une communauté d'adversaires motivés et hostiles.

Ce chapitre présente une technologie plus complexe et plus agressive pour sécuriser les transactions HTTP contre les écoutes clandestines et les falsifications, en utilisant la cryptographie numérique.

Rendre HTTP sûr

Les transactions en ligne sont utilisées pour des raisons importantes. Sans sécurité renforcée, les achats et les opérations bancaires en ligne seraient difficiles à réaliser. Sans possibilité de restreindre l'accès, les entreprises ne pourraient pas stocker de documents importants sur des serveurs web. Le Web nécessite une forme sécurisée de HTTP.

Les chapitres précédents ont présenté des méthodes légères d'authentification (authentification de base et authentification digest) et d'intégrité des messages (digest qop="auth-int"). Ces méthodes sont efficaces à de nombreuses fins, mais peuvent s'avérer insuffisantes pour les achats importants, les transactions bancaires ou l'accès à des données confidentielles. Pour ces transactions plus complexes, nous combinons HTTP et une technologie de chiffrement numérique.

Une version sécurisée de HTTP doit être efficace, portable, facile à administrer et adaptable à un monde en constante évolution. Elle doit également répondre aux exigences sociétales et gouvernementales. Nous avons besoin d'une technologie de sécurité HTTP offrant :

• Authentification du serveur (les clients savent qu'ils parlent au vrai serveur, pas à un faux)

• Authentification du client (les serveurs savent qu'ils parlent au véritable utilisateur, pas à un faux)

• Intégrité (les clients et les serveurs sont protégés contre toute modification de leurs données)

• Cryptage (les clients et les serveurs parlent en privé sans crainte d'être écoutés)

• Efficacité (un algorithme suffisamment rapide pour être utilisé par des clients et des serveurs peu coûteux)

• Ubiquité (les protocoles sont pris en charge par pratiquement tous les clients et serveurs)

• Évolutivité administrative (communication sécurisée instantanée pour n'importe qui, n'importe où)

• Adaptabilité (prend en charge les meilleures méthodes de sécurité connues du moment)

• Viabilité sociale (répond aux besoins culturels et politiques de la société)

HTTPS

HTTPS est la forme sécurisée de HTTP la plus répandue. Elle a été lancée par Netscape.

Communications Corporation et est pris en charge par tous les principaux navigateurs et serveurs.

Vous pouvez savoir si une page Web a été consultée via HTTPS au lieu de HTTP, car l'URL commencera par le schéma https:// au lieu de http:// (certains navigateurs affichent également des signaux de sécurité iconiques, comme illustré dans la Figure 14-1).

Figure 14-1. Navigation sur des sites web sécurisés

Lors de l'utilisation du protocole HTTPS, toutes les données de requête et de réponse HTTP sont chiffrées avant d'être envoyées sur le réseau. Le protocole HTTPS fournit une couche de sécurité cryptographique au niveau du transport, utilisant soit le protocole SSL (Secure Sockets Layer) soit son successeur, le protocole TLS (Transport Layer Security), sous HTTP (Figure 14-2). En raison de la similitude de SSL et de TLS, nous utilisons dans ce livre le terme « SSL » au sens large pour désigner à la fois SSL et TLS.

Étant donné que la plupart des travaux d'encodage et de décodage durs se déroulent dans les bibliothèques SSL, les clients et serveurs Web n'ont pas besoin de modifier beaucoup leur logique de traitement de protocole.

Figure 14-2. HTTPS est un protocole HTTP superposé à une couche de sécurité, superposé à TCP

Pour utiliser le protocole HTTP sécurisé, il suffit généralement de remplacer les appels d'entrée/sortie TCP par des appels SSL et d'ajouter quelques autres appels pour configurer et gérer les informations de sécurité.

Cryptographie numérique

Avant d'aborder le protocole HTTPS en détail, il est important de présenter brièvement les techniques de cryptage cryptographique utilisées par SSL et HTTPS. Dans les sections suivantes, nous vous présenterons rapidement les bases de la cryptographie numérique. Si vous connaissez déjà la technologie et la terminologie de la cryptographie numérique, n'hésitez pas à consulter la section « HTTPS : les détails ».

Dans ce manuel de cryptographie numérique, nous parlerons de :

Chiffres

Algorithmes permettant de coder du texte pour le rendre illisible aux voyeurs

Clés

Paramètres numériques qui modifient le comportement des chiffrements

Cryptosystèmes à clé symétrique

Algorithmes qui utilisent la même clé pour l'encodage et le décodage

Cryptosystèmes à clé asymétrique

Algorithmes qui utilisent des clés différentes pour l'encodage et le décodage

Cryptographie à clé publique

Un système permettant à des millions d'ordinateurs d'envoyer facilement des messages secrets

Signatures numériques

Sommes de contrôle qui vérifient qu'un message n'a pas été falsifié ou falsifié

Certificats numériques

Informations d'identification, vérifiées et signées par un organisme de confiance

Cryptographie numérique

L'art et la science du codage secret

La cryptographie est l'art et la science du codage et du décodage des messages. Depuis des millénaires, l'homme utilise des méthodes cryptographiques pour envoyer des messages secrets. Cependant, la cryptographie ne se limite pas à chiffrer les messages pour empêcher leur lecture par des personnes indiscrètes ; elle peut également servir à empêcher leur falsification. Elle peut même servir à prouver que vous êtes bien l'auteur d'un message ou d'une transaction, tout comme votre signature manuscrite sur un chèque ou un sceau de cire gravé sur une enveloppe.

Chiffres

La cryptographie repose sur des codes secrets appelés chiffrements. Un chiffrement est un schéma de codage : une méthode particulière pour coder un message et une méthode complémentaire pour le décoder ultérieurement. Le message original, avant son codage, est souvent appelé texte clair. Le message codé, après application du chiffrement, est souvent appelé texte chiffré. La figure 14-3 en présente un exemple simple.

Figure 14-3. Texte en clair et texte chiffré

Les chiffrements sont utilisés pour générer des messages secrets depuis des millénaires. La légende raconte que Jules César utilisait un chiffrement à rotation de trois caractères, où chaque caractère du message est remplacé par un caractère trois positions alphabétiques

plus haut. Dans notre alphabet moderne, « A » serait remplacé par « D », « B » par « E », et ainsi de suite.

Par exemple, dans la figure 14-4, le message « rendez-vous à l'embarcadère à minuit » est codé dans le texte chiffré « phhw ph dw wkh slhu dw plgqljkw » en utilisant le chiffrement rot3 (rotation de 3 caractères).* Le texte chiffré peut être déchiffré pour revenir au message en clair d'origine en appliquant le codage inverse, en faisant pivoter de -3 caractères dans l'alphabet.

Figure 14-4. Exemple de chiffrement par rotation de 3

* Pour simplifier l'exemple, nous ne faisons pas tourner la ponctuation ou les espaces, mais vous pourriez le faire.

Machines de chiffrement

Les chiffrements étaient à l'origine des algorithmes relativement simples, car les êtres humains devaient les encoder et les décoder eux-mêmes. Grâce à leur simplicité, les codes pouvaient être déchiffrés avec un crayon, du papier et des livres de codes. Cependant, des personnes intelligentes pouvaient également les déchiffrer assez facilement.

Avec les progrès technologiques, les gens ont commencé à fabriquer des machines capables de coder et de décoder rapidement et précisément des messages grâce à des chiffrements beaucoup plus complexes. Au lieu de se contenter de simples rotations, ces machines pouvaient substituer des caractères, inverser leur ordre et découper les messages pour rendre les codes beaucoup plus difficiles à déchiffrer.*

Chiffres à clé

Comme les algorithmes et les machines pouvaient tomber entre des mains ennemies, la plupart des machines étaient équipées de boutons permettant de régler un grand nombre de valeurs différentes modifiant le fonctionnement du chiffrement. Même en cas de vol, sans les bons réglages (valeurs clés), le décodeur ne fonctionnerait pas.†

Ces paramètres de chiffrement étaient appelés clés. Il fallait saisir la bonne clé dans la machine de chiffrement pour que le processus de décodage fonctionne correctement. Les clés de chiffrement permettent à une machine de chiffrement unique de se comporter comme un ensemble de machines de chiffrement virtuelles, chacune se comportant différemment selon ses valeurs de clé.

La figure 14-5 illustre un exemple de chiffrement à clé. L'algorithme de chiffrement utilisé est le chiffrement trivial « rotation par N ». La valeur de N est contrôlée par la clé. Le même message d'entrée, « Rendez-vous à l'embarcadère à minuit », transmis par la même machine de chiffrement, génère des sorties différentes selon la valeur de la clé.

Aujourd'hui, la quasi-totalité des algorithmes de chiffrement utilisent des clés.

Chiffres numériques

Avec l’avènement du calcul numérique, deux avancées majeures ont eu lieu :

• Des algorithmes de codage et de décodage complexes sont devenus possibles, libérés des limitations de vitesse et de fonctionnalité des machines mécaniques.

* La machine à code mécanique la plus célèbre fut sans doute la machine à coder allemande Enigma de la Seconde Guerre mondiale. Malgré la complexité du chiffrement Enigma, Alan Turing et ses collègues parvinrent à déchiffrer les codes Enigma au début des années 1940, grâce aux premiers ordinateurs numériques.

† En réalité, posséder la logique de la machine peut parfois aider à déchiffrer le code, car elle peut indiquer des schémas exploitables. Les algorithmes cryptographiques modernes sont généralement conçus de telle sorte que, même s'ils sont connus du public, il est difficile de trouver des schémas permettant aux malfaiteurs de les déchiffrer. En fait, le code source de la plupart des chiffrements les plus puissants et couramment utilisés est accessible au public, accessible à tous !

Cryptographie numérique

Figure 14-5. Chiffrement par rotation par N, utilisant différentes clés

• Il est devenu possible de prendre en charge des clés très grandes, de sorte qu'un seul algorithme de chiffrement pouvait produire des milliards d'algorithmes de chiffrement virtuels, chacun différant par la valeur de la clé. Plus la clé est longue, plus les combinaisons de codages possibles sont nombreuses et plus il est difficile de déchiffrer le code en devinant des clés aléatoires.

Contrairement aux touches métalliques physiques ou aux molettes des appareils mécaniques, les touches numériques ne sont que des chiffres. Ces valeurs numériques servent d'entrées aux algorithmes de codage et de décodage. Ces algorithmes sont des fonctions qui prennent un bloc de données et le codent/le décodent en fonction de l'algorithme et de la valeur de la touche.

Étant donné un message en clair appelé P, une fonction de codage appelée E et une clé de codage numérique appelée e, vous pouvez générer un message chiffré codé C (Figure 14-6). Vous pouvez décoder le texte chiffré C en texte clair original P en utilisant la fonction de décodage D et la clé de décodage d. Bien entendu, les fonctions de décodage et de codage sont inverses ; le décodage du codage de P restitue le message original P.

Figure 14-6. Le texte en clair est codé avec la clé de codage e et décodé avec la clé de décodage d.

Cryptographie à clé symétrique

Examinons plus en détail la façon dont les clés et les chiffrements fonctionnent ensemble. De nombreux algorithmes de chiffrement numérique sont appelés chiffrements à clé symétrique, car ils utilisent la même valeur de clé pour le codage et le décodage (e = d). Appelons simplement la clé k.

Dans un chiffrement à clé symétrique, l'expéditeur et le destinataire doivent partager la même clé secrète, k, pour communiquer. L'expéditeur utilise cette clé secrète pour chiffrer le message et envoie le texte chiffré obtenu au destinataire. Ce dernier récupère le texte chiffré et applique la fonction de déchiffrement, avec la même clé secrète partagée, pour récupérer le texte en clair d'origine (Figure 14-7).

Figure 14-7. Les algorithmes de cryptographie à clé symétrique utilisent la même clé pour le codage et le décodage.

Certains algorithmes de chiffrement à clé symétrique populaires sont DES, Triple-DES, RC2 et RC4.

Attaques par longueur de clé et par énumération

Il est essentiel que les clés secrètes restent secrètes. Dans la plupart des cas, les algorithmes de codage et de décodage sont publics ; la clé est donc la seule chose secrète !

Un bon algorithme de chiffrement force l'ennemi à essayer toutes les clés possibles pour déchiffrer le code. Essayer toutes les clés par force brute est appelé une attaque par énumération. S'il n'y a que quelques clés possibles, un pirate peut les explorer toutes par force brute et finalement déchiffrer le code. En revanche, s'il y en a beaucoup, il lui faudra peut-être des jours, des années, voire l'éternité, pour explorer toutes les clés et trouver celle qui déchiffre le chiffrement.

Cryptographie à clé symétrique

Le nombre de valeurs de clé possibles dépend du nombre de bits de la clé et du nombre de clés valides. Pour les chiffrements à clé symétrique, toutes les valeurs de clé sont généralement valides.* Une clé de 8 bits n'aurait que 256 clés possibles, une clé de 40 bits en aurait 240 (environ mille milliards de clés) et une clé de 128 bits générerait environ 340 000 000 000 000 000 000 000 000 000 000 000 000 000.

Pour les chiffrements à clé symétrique conventionnels, les clés de 40 bits sont considérées comme suffisamment sûres pour les petites

transactions non critiques. Cependant, elles sont vulnérables aux stations de travail ultra-rapides actuelles, capables d'effectuer des milliards de calculs par seconde.

En revanche, les clés de 128 bits sont considérées comme très robustes pour la cryptographie à clé symétrique. En fait, les clés longues ont un tel impact sur la sécurité cryptographique que le gouvernement américain a instauré des contrôles à l'exportation sur les logiciels cryptographiques utilisant des clés longues, afin d'empêcher des organisations potentiellement hostiles de créer des codes secrets que l'Agence nationale de sécurité américaine (NSA) serait elle-même incapable de déchiffrer.

L'excellent livre de Bruce Schneier, Applied Cryptography (John Wiley & Sons), comprend un tableau décrivant le temps qu'il faudrait pour déchiffrer un chiffrement DES en devinant toutes les clés, en utilisant la technologie et l'économie de 1995.† Des extraits de ce tableau sont présentés dans le tableau 14-1.

Tableau 14-1. Les clés longues nécessitent plus d'efforts pour être déchiffrées (données de 1995, tirées de « Cryptographie appliquée »).

Coût d'attaque Clé 40 bits Clé 56 bits Clé 64 bits Clé 80 bits Clé 128 bits

100 000 $ 2 secondes 35 heures 1 an 70 000 ans 1019 ans

1 000 000 $ 200 ms 3,5 heures 37 jours 7 000 ans 1018 ans

10 000 000 $ 20 ms 21 min 4 jours 700 ans 1017 ans

100 000 000 $ 2 ms 2 min 9 heures 70 ans 1016 ans

1 000 000 000 $ 200 usecs 13 secondes 1 heure 7 ans 1015 ans

Compte tenu de la rapidité des microprocesseurs de 1995, un attaquant prêt à dépenser 100 000 dollars en 1995 pouvait déchiffrer un code DES de 40 bits en environ 2 secondes. Or, les ordinateurs de 2002 sont déjà 20 fois plus rapides qu'en 1995. À moins que les utilisateurs ne changent fréquemment de clés, les clés de 40 bits ne sont pas sûres contre des adversaires motivés.

La taille de clé standard DES de 56 bits est plus sûre. En 1995, une attaque d'un million de dollars nécessiterait encore plusieurs heures pour déchiffrer le code. Mais une personne ayant accès à des superordinateurs pourrait le déchiffrer par force brute en quelques secondes.

* Il existe des chiffrements où seules certaines valeurs de clé sont valides. Par exemple, dans RSA, le système de chiffrement à clé asymétrique le plus connu, les clés valides doivent être liées à des nombres premiers d'une certaine manière. Seul un petit nombre de valeurs de clé possibles présentent cette propriété.

† La vitesse de calcul a considérablement augmenté depuis 1995, et les coûts ont diminué. Et plus vous mettrez de temps à lire ce livre, plus ils deviendront rapides ! Cependant, le tableau reste relativement utile, même si les temps sont décalés d'un facteur 5, 10, voire plus.

En revanche, les clés DES 128 bits, de taille similaire aux clés Triple-DES, sont censées être

effectivement incassable par quiconque, à n'importe quel prix, en utilisant une attaque par force brute.*

Établissement de clés partagées

L'un des inconvénients des chiffrements à clé symétrique est que l'expéditeur et le destinataire doivent tous deux avoir une clé secrète partagée avant de pouvoir communiquer entre eux.

Si vous vouliez communiquer en toute sécurité avec Joe's Hardware, par exemple pour commander des outils de menuiserie après avoir regardé une émission de bricolage à la télévision publique, vous devriez établir une clé secrète privée entre vous et www.joes-hardware.com avant de pouvoir commander quoi que ce soit en toute sécurité. Il vous faudrait un moyen de générer cette clé secrète et de vous en souvenir. Vous, Joe's Hardware et tous les autres internautes auriez des milliers de clés à générer et à mémoriser.

Imaginons qu'Alice (A), Bob (B) et Chris (C) souhaitent tous communiquer avec Joe's Hardware (J). A, B et C doivent chacun établir leur propre clé secrète avec J. A a besoin de la clé kAJ, B de la clé kBJ et C de la clé kCJ. Chaque paire de parties communicantes a besoin de sa propre clé privée. S'il y a N nœuds, et que chaque nœud doit communiquer de manière sécurisée avec les N–1 autres nœuds, il y a environ N2 clés secrètes au total : un véritable cauchemar administratif.

Cryptographie à clé publique

Au lieu d'une clé de codage/décodage unique pour chaque paire d'hôtes, la cryptographie à clé publique utilise deux clés asymétriques : une pour coder les messages d'un hôte, et une autre pour les décoder. La clé de codage est publiquement connue (d'où le nom de cryptographie à clé publique), mais seul l'hôte connaît la clé privée de décodage (voir Figure 14-8). Cela facilite grandement l'établissement des clés, car chacun peut trouver la clé publique d'un hôte particulier. En revanche, la clé de décodage est gardée secrète, de sorte que seul le destinataire peut décoder les messages qui lui sont adressés.

Le nœud X peut prendre sa clé de chiffrement ex et la publier publiquement.† Désormais, toute personne souhaitant envoyer un message au nœud X peut utiliser la même clé publique, bien connue. Étant donné que chaque hôte se voit attribuer une clé de chiffrement, que tout le monde utilise, la cryptographie à clé publique évite l'explosion N2 des clés symétriques par paires (voir Figure 14-9).

* Une clé de grande taille ne garantit pas pour autant l'infaillibilité du chiffrement ! Il peut y avoir une faille non détectée dans l'algorithme ou l'implémentation du chiffrement, qui constitue une faiblesse exploitable par un attaquant. Il est également possible que l'attaquant dispose d'informations sur la manière dont les clés sont générées, ce qui lui permet de savoir que certaines clés sont plus probables que d'autres, ce qui favorise une attaque par force brute. Un utilisateur peut également laisser la clé secrète à un endroit où un attaquant pourrait la voler.

† Comme nous le verrons plus tard, la plupart des recherches de clés publiques sont en fait effectuées via des certificats numériques, mais les détails de la façon dont vous trouvez les clés publiques n'ont plus beaucoup d'importance maintenant : sachez simplement qu'elles sont accessibles au public quelque part.

Cryptographie à clé publique

Figure 14-9. La cryptographie à clé publique attribue une clé de chiffrement publique unique à chaque hôte.

Même si tout le monde peut encoder des messages vers X avec la même clé, seul X peut les décoder, car seul X possède la clé privée de décodage dx. La séparation des clés permet à chacun d'encoder un message, mais limite la capacité de décodage des messages au seul propriétaire. Cela facilite l'envoi sécurisé de messages aux serveurs par les nœuds, car ils peuvent simplement consulter la clé publique du serveur.

La technologie de chiffrement à clé publique permet de déployer des protocoles de sécurité pour tous les utilisateurs d'ordinateurs dans le monde. Compte tenu de l'importance cruciale de la standardisation d'une suite technologique de clés publiques, un vaste projet de normalisation des infrastructures à clés publiques (ICP) est en cours depuis plus de dix ans.

RSA

Le défi de tout cryptosystème asymétrique à clé publique est de s'assurer qu'aucun malfaiteur ne puisse calculer la clé privée secrète, même s'il dispose de tous les indices suivants :

• La clé publique (que tout le monde peut obtenir, car elle est publique)

• Un morceau de texte chiffré intercepté (obtenu en fouinant le réseau)

• Un message et son texte chiffré associé (obtenu en exécutant l'encodeur sur n'importe quel

texte)

L'algorithme RSA, inventé au MIT et commercialisé par RSA Data Security, est un système de cryptographie à clé publique populaire répondant à tous ces besoins. Étant donné une clé publique, un fragment arbitraire de texte en clair, le texte chiffré associé issu du codage du texte en clair avec la clé publique, l'algorithme RSA lui-même et même le code source de son implémentation, déchiffrer le code pour trouver la clé privée correspondante est considéré comme un problème aussi difficile que le calcul de grands nombres premiers – considéré comme l'un des plus complexes de toute l'informatique. Ainsi, si vous parvenez à factoriser rapidement de grands nombres en nombres premiers, non seulement vous pourrez accéder à des comptes bancaires suisses, mais vous pourrez également remporter un prix Turing.

Les détails de la cryptographie RSA impliquent des mathématiques complexes ; nous ne les aborderons donc pas ici. De nombreuses bibliothèques permettent d'exécuter les algorithmes RSA sans avoir besoin d'un doctorat en théorie des nombres.

Cryptosystèmes hybrides et clés de session

La cryptographie asymétrique à clé publique est astucieuse, car n'importe qui peut envoyer des messages sécurisés à un serveur public, simplement en connaissant sa clé publique. Deux nœuds n'ont pas besoin de négocier au préalable une clé privée pour communiquer de manière sécurisée.

Cependant, les algorithmes de cryptographie à clé publique ont tendance à être lents en termes de calcul. En pratique, on utilise des combinaisons de schémas symétriques et asymétriques. Par exemple, il est courant d'utiliser la cryptographie à clé publique pour établir facilement une communication sécurisée entre les nœuds, puis d'utiliser ce canal sécurisé pour générer et communiquer une clé symétrique aléatoire temporaire afin de chiffrer le reste des données grâce à une cryptographie symétrique plus rapide.

Signatures numériques

Jusqu'à présent, nous avons parlé de différents types de chiffrements à clé, utilisant des clés symétriques et asymétriques, pour nous permettre de crypter et de décrypter des messages secrets.

Signatures numériques

Outre le chiffrement et le déchiffrement des messages, les systèmes de chiffrement permettent de signer des messages, prouvant ainsi leur auteur et leur intégrité. Cette technique, appelée signature numérique, est importante pour les certificats de sécurité Internet, que nous aborderons dans la section suivante.

Les signatures sont des sommes de contrôle cryptographiques

Les signatures numériques sont des sommes de contrôle cryptographiques spécifiques attachées à un message. Elles présentent deux avantages :

• Les signatures prouvent que l'auteur a écrit le message. Étant donné que seul l'auteur possède sa clé privée top secrète*, il est le seul à pouvoir calculer ces sommes de contrôle. La somme de contrôle fait office de signature personnelle de l'auteur.

• Les signatures empêchent la falsification des messages. Si un attaquant malveillant modifiait le message en cours de transmission, la somme de contrôle ne correspondrait plus. Et comme cette somme de contrôle implique la clé privée secrète de l'auteur, l'intrus ne pourrait pas fabriquer une somme de contrôle correcte pour le message falsifié.

Les signatures numériques sont souvent générées à l'aide d'une technologie à clé publique asymétrique. La clé privée de l'auteur est utilisée comme une sorte d'« empreinte digitale », car elle n'est connue que de son propriétaire.

La figure 14-10 montre un exemple de la manière dont le nœud A peut envoyer un message au nœud B et le signer :

• Le nœud A distille le message de longueur variable en un condensé de taille fixe.

• Le nœud A applique une fonction de « signature » au condensé, utilisant la clé privée de l'utilisateur comme paramètre. Étant donné que seul l'utilisateur connaît la clé privée, une fonction de signature correcte indique que le signataire est le propriétaire. Dans la figure 14-10, nous utilisons le décodeur.

fonction D comme fonction de signature, car elle implique la clé privée de l'utilisateur.†

• Une fois la signature calculée, le nœud A l'ajoute à la fin du message et envoie à la fois le message et la signature au nœud B.

• À la réception, si le nœud B souhaite s'assurer que le nœud A a bien écrit le message et qu'il n'a pas été falsifié, il peut vérifier la signature. Il prend la signature brouillée par clé privée et applique la fonction inverse en utilisant la clé publique. Si le résumé décompressé ne correspond pas à la version du résumé du nœud B, soit le message a été falsifié en cours de route, soit l'expéditeur ne disposait pas de la clé privée du nœud A (et n'était donc pas le nœud A).

* Ceci suppose que la clé privée n'a pas été volée. La plupart des clés privées expirent après un certain temps. Il existe également des « listes de révocation » qui permettent de suivre les clés volées ou compromises.

† Avec le cryptosystème RSA, la fonction de décodage D est utilisée comme fonction de signature, car D prend déjà la clé privée en entrée. Notez que la fonction de décodage n'est qu'une fonction ; elle peut donc être utilisée sur n'importe quelle entrée. De plus, dans le cryptosystème RSA, les fonctions D et E fonctionnent indépendamment l'une de l'autre et s'annulent. Ainsi, E(D(stuff)) = stuff, tout comme D(E(stuff)) = stuff.

Figure 14-10. Signature numérique non chiffrée

Certificats numériques

Dans cette section, nous abordons les certificats numériques, véritables « cartes d'identité » d'Internet. Les certificats numériques (souvent appelés « certs », comme les pastilles à la menthe) contiennent des informations sur un utilisateur ou une entreprise, certifiées par une organisation de confiance.

Nous possédons tous de nombreux documents d'identité. Certains, comme le passeport et le permis de conduire, sont suffisamment fiables pour prouver notre identité dans de nombreuses situations. Par exemple, un permis de conduire américain suffit pour vous permettre de monter à bord d'un avion à destination de New York pour le réveillon du Nouvel An, et pour vous permettre de boire des boissons alcoolisées avec vos amis à votre arrivée.

Les pièces d'identité les plus fiables, comme les passeports, sont signées et tamponnées par un gouvernement sur un papier spécial. Plus difficiles à falsifier, elles bénéficient d'un niveau de confiance plus élevé. Certains badges et cartes à puce d'entreprise intègrent des composants électroniques pour renforcer l'identité du porteur. Certaines organisations gouvernementales ultra-secrètes ont même besoin de comparer vos empreintes digitales ou vos tracés capillaires rétiniens avec votre identité.

Identifiez-le avant de lui faire confiance !

D'autres pièces d'identité, comme les cartes de visite, sont relativement faciles à falsifier, ce qui rend les gens moins confiants. Elles peuvent convenir pour des interactions professionnelles, mais ne constituent probablement pas une preuve d'emploi suffisante pour une demande de prêt immobilier.

Les entrailles d'un certificat

Les certificats numériques contiennent également un ensemble d'informations, toutes signées numériquement par une « autorité de certification » officielle. Les certificats numériques de base contiennent généralement des éléments communs aux pièces d'identité imprimées, tels que :

• Nom du sujet (personne, serveur, organisation, etc.)

• Date d'expiration

Certificats numériques

• Émetteur du certificat (qui se porte garant du certificat)

• Signature numérique de l’émetteur du certificat

De plus, les certificats numériques contiennent souvent la clé publique du sujet, ainsi que des informations descriptives sur ce sujet et sur l'algorithme de signature utilisé. N'importe qui peut créer un certificat numérique, mais tout le monde ne peut pas obtenir d'une autorité de signature reconnue qu'elle garantisse les informations du certificat et le signe avec sa clé privée. Une structure de certificat typique est illustrée à la figure 14-11.

Figure 14-11. Format typique de signature numérique

Certificats X.509 v3

Malheureusement, il n'existe pas de norme unique et universelle pour les certificats numériques. Il existe de nombreux styles de certificats numériques, subtilement différents, tout comme les cartes d'identité imprimées ne contiennent pas toutes les mêmes informations au même endroit. Heureusement, la plupart des certificats utilisés aujourd'hui stockent leurs informations sous une forme standard, appelée X.509 v3. Les certificats X.509 v3 offrent une méthode standardisée pour structurer les informations de certificat en champs analysables. Les différents types de certificats ont des valeurs de champ différentes, mais la plupart suivent la structure X.509 v3. Les champs d'un certificat X.509 sont décrits dans le tableau 14-2.

Tableau 14-2. Champs du certificat X.509

Description du champ

Version : numéro de version du certificat X.509. Actuellement, il s'agit généralement de la version 3.

Numéro de série : un entier unique généré par l'autorité de certification. Chaque certificat d'une autorité de certification doit avoir un numéro de série unique.

ID de l'algorithme de signature : algorithme cryptographique utilisé pour la signature. Par exemple, « MD2 digest avec chiffrement RSA ».

Émetteur du certificat Le nom de l'organisation qui a émis et signé ce certificat, au format X.500.

Période de validité Pendant laquelle ce certificat est valide, défini par une date de début et une date de fin.

Tableau 14-2. Champs du certificat X.509 (suite)

Description du champ

Nom du sujet : l’entité décrite dans le certificat, telle qu’une personne ou une organisation. Le nom du sujet est au format X.500.

Informations sur la clé publique du sujet La clé publique du sujet du certificat, l'algorithme utilisé pour la clé publique et tous les paramètres supplémentaires.

ID unique de l'émetteur (facultatif) Un identifiant unique facultatif pour l'émetteur du certificat, pour permettre la réutilisation potentielle du même nom d'émetteur.

ID unique du sujet (facultatif) Un identifiant unique facultatif pour le sujet du certificat, pour permettre la réutilisation potentielle du même nom de sujet.

Extensions : un ensemble facultatif de champs d'extension (version 3 et ultérieures). Chaque champ d'extension est signalé comme critique ou non critique. Les extensions critiques sont importantes et doivent être comprises par l'utilisateur du certificat. Si un utilisateur ne reconnaît pas un champ d'extension critique, il doit rejeter le certificat. Les champs d'extension couramment utilisés sont les suivants :

Contraintes de base

Relation du sujet avec l'autorité de certification

Politique de certification

La politique en vertu de laquelle le certificat est accordé

Utilisation des clés

Restreint la manière dont la clé publique peut être utilisée

Signature de l'autorité de certification La signature numérique de l'autorité de certification de tous les champs ci-dessus, en utilisant l'algorithme de signature spécifié.

Il existe plusieurs types de certificats basés sur X.509, notamment (entre autres) les certificats de serveur Web, les certificats de messagerie client, les certificats de signature de code logiciel et les certificats d'autorité de certification.

Utilisation de certificats pour authentifier les serveurs

Lors d'une transaction web sécurisée via HTTPS, les navigateurs modernes récupèrent automatiquement le certificat numérique du serveur auquel vous vous connectez. Si le serveur ne possède pas de certificat, la connexion sécurisée échoue. Le certificat du serveur contient de nombreux champs, notamment :

• Nom et nom d'hôte du site Web

• Clé publique du site Web

• Nom du signataire autorisé

• Signature du signataire autorisé

Lorsque le navigateur reçoit le certificat, il vérifie l'autorité de signature.* S'il s'agit d'une autorité de signature publique et respectée, le navigateur connaîtra déjà sa clé publique

* Les navigateurs et autres applications Internet s'efforcent de masquer les détails de la gestion des certificats afin de faciliter la navigation. Cependant, lorsque vous naviguez via des connexions sécurisées, tous les principaux navigateurs vous permettent d'examiner personnellement les certificats des sites auxquels vous communiquez, afin de vous assurer que tout est conforme.

Certificats numériques

(les navigateurs sont livrés avec des certificats de nombreuses autorités de signature préinstallés), afin de pouvoir vérifier la signature comme nous l'avons vu dans la section précédente, « Signatures numériques ».

La figure 14-12 montre comment l’intégrité d’un certificat est vérifiée à l’aide de sa signature numérique.

Figure 14-12. Vérification de l'authenticité d'une signature

Si l'autorité de signature est inconnue, le navigateur ne sait pas s'il doit lui faire confiance et affiche généralement une boîte de dialogue permettant à l'utilisateur de vérifier s'il fait confiance au signataire. Le signataire peut être le service informatique local ou un éditeur de logiciels.

HTTPS : les détails

HTTPS est la version sécurisée la plus populaire de HTTP. Largement implémentée et disponible sur tous les principaux navigateurs et serveurs commerciaux, elle combine le protocole HTTP avec un puissant ensemble de techniques cryptographiques symétriques, asymétriques et basées sur des certificats, ce qui la rend très sécurisée, mais aussi très flexible et facile à administrer dans l'anarchie d'un Internet mondial décentralisé.

HTTPS a accéléré la croissance des applications Internet et a joué un rôle majeur dans la croissance rapide du commerce électronique en ligne. HTTPS a également joué un rôle essentiel dans l'administration sécurisée et étendue des applications web distribuées.

Présentation de HTTPS

HTTPS est simplement du HTTP envoyé via une couche de transport sécurisée. Au lieu d'envoyer des messages HTTP non chiffrés à TCP et

sur Internet (figure 14-13a), HTTPS les transmet d'abord à une couche de sécurité qui les chiffre avant de les transmettre à TCP (figure 14-13b).

Figure 14-13. Sécurité au niveau du transport HTTP

Aujourd'hui, la couche de sécurité HTTP est assurée par SSL et son remplaçant moderne, TLS. Nous utilisons couramment le terme « SSL » pour désigner soit SSL, soit TLS.

Schémas HTTPS

Aujourd'hui, le protocole HTTP sécurisé est facultatif. Ainsi, lors d'une requête adressée à un serveur web, nous devons pouvoir lui indiquer d'exécuter la version sécurisée du protocole HTTP.

Cela se fait dans le schéma de l'URL.

Dans un protocole HTTP normal et non sécurisé, le préfixe de schéma de l'URL est http, comme dans :

http://www.joes-hardware.com/index.html

Dans le protocole sécurisé HTTPS, le préfixe de schéma de l'URL est https, comme dans :

https://cajun-shop.securesites.com/Merchant2/merchant.mv?Store_Code=AGCGS

Lorsqu'un client (tel qu'un navigateur Web) est invité à effectuer une transaction sur une ressource Web, il examine le schéma de l'URL :

• Si l'URL a un schéma http, le client ouvre une connexion au serveur sur le port 80 (par défaut) et lui envoie des commandes HTTP simples (Figure 14-14a).

• Si l'URL a un schéma https, le client ouvre une connexion au serveur sur le port 443 (par défaut) puis « établit une liaison » avec le serveur, échangeant certains paramètres de sécurité SSL avec le serveur dans un format binaire, suivis des commandes HTTP cryptées (Figure 14-14b).

Le trafic SSL étant un protocole binaire, totalement différent de HTTP, il est acheminé sur des ports différents (SSL est généralement acheminé sur le port 443). Si le trafic SSL et HTTP arrivait sur le port 80, la plupart des serveurs web interpréteraient le trafic SSL binaire comme un HTTP erroné et fermeraient la connexion. Une intégration plus poussée des services de sécurité dans HTTP aurait éliminé la nécessité de plusieurs ports de destination, mais cela ne pose pas de problèmes majeurs en pratique.

Examinons de plus près comment SSL établit des connexions avec des serveurs sécurisés.

Figure 14-14. Numéros de port HTTP et HTTPS

Configuration du transport sécurisé

En HTTP non chiffré, un client ouvre une connexion TCP sur le port 80 d'un serveur web, envoie un message de requête, reçoit un message de réponse et ferme la connexion. Cette séquence est illustrée à la figure 14-15a.

La procédure est légèrement plus complexe en HTTPS, en raison de la couche de sécurité SSL. En HTTPS, le client ouvre d'abord une connexion sur le port 443 (port par défaut pour le HTTP sécurisé) du serveur web. Une fois la connexion TCP établie, le client et le serveur initialisent la couche SSL, négociant les paramètres de cryptographie et échangeant les clés. Une fois la négociation terminée, l'initialisation SSL est terminée et le client peut envoyer des messages de requête à la couche de sécurité. Ces messages sont chiffrés avant d'être envoyés à TCP. Cette procédure est illustrée à la figure 14-15b.

Poignée de main SSL

Avant de pouvoir envoyer des messages HTTP chiffrés, le client et le serveur doivent effectuer une négociation SSL, où ils :

• Numéros de version du protocole d'échange

• Sélectionnez un chiffrement que chaque partie connaît

• Authentifier l’identité de chaque partie

• Générer des clés de session temporaires pour crypter le canal

Figure 14-15. Transactions HTTP et HTTPS

Avant que les données HTTP chiffrées ne transitent sur le réseau, SSL a déjà envoyé un ensemble de données de négociation pour établir la communication. Le principe de la négociation SSL est illustré à la figure 14-16.

Il s'agit d'une version simplifiée de la négociation SSL. Selon la manière dont SSL est utilisé, la négociation peut être plus complexe, mais voici l'idée générale.

Figure 14-16. Poignée de main SSL (simplifiée)

Certificats de serveur

SSL prend en charge l'authentification mutuelle, transmettant les certificats serveur aux clients et les retournant aux serveurs. Cependant, aujourd'hui, les certificats clients sont peu utilisés pour la

navigation. La plupart des utilisateurs ne possèdent même pas de certificat client personnel*. Un serveur web peut exiger un certificat client, mais cela se produit rarement en pratique†.

En revanche, les transactions HTTPS sécurisées nécessitent toujours des certificats de serveur. Lorsque vous effectuez une transaction sécurisée sur un serveur web, comme la publication de vos informations de carte de crédit, vous souhaitez être sûr de communiquer avec l'organisation à laquelle vous pensez vous adresser. Les certificats de serveur, signés par une autorité reconnue, vous aident à évaluer votre degré de confiance envers le serveur avant de transmettre vos informations de carte de crédit ou vos informations personnelles.

Le certificat du serveur est un certificat dérivé de X.509 v3 indiquant le nom, l'adresse, le nom de domaine DNS du serveur et d'autres informations de l'organisation (voir la figure 14-17). Vous et votre logiciel client pouvez examiner le certificat pour vous assurer que tout est en ordre.

* Les certificats clients sont utilisés pour la navigation web dans certaines entreprises, et pour la messagerie sécurisée. À l'avenir, ils pourraient se généraliser pour la navigation web, mais leur adoption est encore lente.

† Certains intranets organisationnels utilisent des certificats clients pour contrôler l’accès des employés aux informations.

Figure 14-17. Les certificats HTTPS sont des certificats X.509 contenant des informations sur le site.

Validation du certificat du site

SSL lui-même ne nécessite pas l'examen du certificat du serveur web, mais la plupart des navigateurs modernes effectuent quelques vérifications simples de leur validité et permettent des vérifications plus approfondies. Un algorithme de validation des certificats de serveur web, proposé par Netscape, constitue la base de la plupart des techniques de validation des navigateurs. Les étapes sont les suivantes :

Vérification des dates

Tout d'abord, le navigateur vérifie les dates de début et de fin du certificat pour s'assurer qu'il est toujours valide. Si le certificat a expiré ou n'est pas encore actif, la validation échoue et le navigateur affiche une erreur.

Vérification de la confiance du signataire

Chaque certificat est signé par une autorité de certification (AC), qui se porte garante du serveur. Il existe différents niveaux de certificat, chacun nécessitant un niveau de vérification des antécédents différent. Par exemple, si vous demandez un certificat de serveur de commerce électronique, vous devez généralement fournir une preuve légale de constitution en société.

N'importe qui peut générer des certificats, mais certaines autorités de certification sont des organisations reconnues disposant de procédures bien établies pour vérifier l'identité et le bon comportement commercial des demandeurs de certificats. C'est pourquoi les navigateurs proposent une liste d'autorités de signature de confiance. Si un navigateur reçoit un certificat signé par une autorité inconnue (et potentiellement malveillante), il affiche généralement un avertissement. Les navigateurs peuvent également choisir d'accepter tout certificat dont le chemin de signature vers une autorité de certification de confiance est valide. Autrement dit, si une autorité de certification de confiance signe un certificat pour « Sam's Signing Shop » et que Sam's Signing Shop signe un certificat de site, le navigateur peut accepter le certificat comme provenant d'un chemin d'accès valide vers une autorité de certification.

Vérification de signature

Une fois que l'autorité de signature est jugée digne de confiance, le navigateur vérifie l'intégrité du certificat en appliquant la clé publique de l'autorité de signature à la signature et en la comparant à la somme de contrôle.

Vérification de l'identité du site

Pour empêcher un serveur de copier le certificat d'un autre serveur ou d'intercepter son trafic, la plupart des navigateurs vérifient que le nom de domaine du certificat correspond à celui du serveur auquel ils ont communiqué. Les certificats de serveur contiennent généralement un seul nom de domaine, mais certaines autorités de certification créent des certificats contenant des listes de noms de serveurs ou des noms de domaine génériques, pour des clusters ou des fermes de serveurs. Si le nom d'hôte ne correspond pas à l'identité du certificat, les clients orientés utilisateur doivent soit avertir l'utilisateur, soit interrompre la connexion avec une erreur de certificat incorrect.

Hébergement virtuel et certificats

Il est parfois difficile de gérer un trafic sécurisé sur des sites hébergés virtuellement (plusieurs noms d'hôtes sur un même serveur). Certains programmes de serveur web populaires ne prennent en charge qu'un seul certificat. Si un utilisateur accède à un nom d'hôte virtuel qui ne

correspond pas exactement au nom du certificat, un message d'avertissement s'affiche.

Prenons l'exemple du site de commerce électronique Cajun-Shop.com, basé sur la Louisiane. L'hébergeur du site a fourni le nom officiel cajun-shop.securesites.com. Lorsque les utilisateurs se rendent sur https://www.cajun-shop.com, le nom d'hôte officiel indiqué dans le certificat du serveur (*.securesites.com) ne correspond pas au nom d'hôte virtuel consulté (www.cajun-shop.com), et l'avertissement de la figure 14-18 s'affiche.

Pour éviter ce problème, les propriétaires de Cajun-Shop.com redirigent tous les utilisateurs vers cajunshop.securesites.com lorsqu'ils commencent des transactions sécurisées. La gestion des certificats pour les sites hébergés virtuellement peut s'avérer complexe.

Un vrai client HTTPS

SSL est un protocole binaire complexe. À moins d'être un expert en cryptographie, il est déconseillé d'envoyer directement du trafic SSL brut. Heureusement, plusieurs bibliothèques commerciales et open source facilitent la programmation des clients et serveurs SSL.

OpenSSL

OpenSSL est l'implémentation open source la plus populaire de SSL et TLS. Le projet OpenSSL est un projet collaboratif et bénévole visant à développer une boîte à outils robuste, commerciale et complète, implémentant les protocoles SSL et TLS, ainsi qu'une bibliothèque de cryptographie polyvalente et performante. Vous pouvez obtenir des informations sur OpenSSL et télécharger le logiciel sur http://www.openssl.org.

Figure 14-18. Les incohérences de nom de certificat entraînent l'affichage de boîtes de dialogue d'erreur.

Vous avez peut-être aussi entendu parler de SSLeay (prononcé SSLeay). OpenSSL est le successeur de la bibliothèque SSLeay et possède une interface très similaire. SSLeay a été initialement développé par Eric A. Young (le « eay » de SSLeay).

Un client HTTPS simple

Dans cette section, nous utiliserons le package OpenSSL pour écrire un client HTTPS extrêmement primitif. Ce client établit une connexion SSL avec un serveur, imprime des informations d'identification du serveur du site, envoie une requête HTTP GET via le canal sécurisé, reçoit une réponse HTTP et l'imprime.

Le programme C ci-dessous est une implémentation OpenSSL du client HTTPS trivial. Pour simplifier le programme, la logique de gestion des erreurs et de traitement des certificats n'a pas été incluse.

La gestion des erreurs ayant été supprimée de cet exemple de programme, son utilisation est strictement réservée à des fins explicatives. En cas d'erreur normale, le logiciel plantera ou se comportera mal.

```
/*************************************************************
**************

* https_client.c --- client HTTPS très simple sans vérification d'erreur

* utilisation : https_client nom_serveur

*************************************************************
************/

#include <stdio.h>

#include <memory.h>

#include <errno.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <netdb.h>

#include <openssl/crypto.h>

#include <openssl/x509.h>

#include <openssl/pem.h>

#include <openssl/ssl.h> #include <openssl/err.h>

void main(int argc, char **argv)

{

   SSL *ssl;

   SSL_CTX *ctx;

   SSL_METHOD *client_method;

   X509 *server_cert; int sd,err; char
*str,*hostname,outbuf[4096],inbuf[4096],host_header[512]; struct
hostent *host_entry; struct sockaddr_in server_socket_address; struct
in_addr ip;

   /*=======================================*/
```

```
   /* (1) initialiser la bibliothèque SSL */

   /*=========================================*/

   SSLeay_add_ssl_algorithms(); client_method =
SSLv2_client_method(); SSL_load_error_strings(); ctx =
SSL_CTX_new(client_method); printf("(1) Contexte SSL initialisé\n\n");

   /*=================================================*/

   /* (2) convertir le nom d'hôte du serveur en adresse IP */
/*=========================================*/

   nom d'hôte = argv[1]; host_entry = gethostbyname(nom d'hôte);

   bcopy(host_entry->h_addr, &(ip.s_addr), host_entry->h_length);
printf("(2) '%s' a l'adresse IP '%s'\n\n", hostname, inet_ntoa(ip));

   /*=================================================*/

   /* (3) ouvrir une connexion TCP sur le port 443 sur le serveur */
/*=================================================*/ sd
= socket (AF_INET, SOCK_STREAM, 0);

   memset(&adresse_socket_serveur, '\0',
sizeof(adresse_socket_serveur)); adresse_socket_serveur.sin_family =
AF_INET; adresse_socket_serveur.sin_port = htons(443);
memcpy(&(adresse_socket_serveur.sin_addr.s_addr), entrée_hôte-
>h_addr, entrée_hôte->h_length);

   err = connect(sd, (struct sockaddr*) &adresse_socket_serveur,
sizeof(adresse_socket_serveur));

   si (err < 0) { perror("impossible de se connecter au port du serveur");
exit(1); }

   printf("(3) Connexion TCP ouverte sur l'hôte '%s', port %d\n\n",
hostname, server_socket_address.sin_port);


/*=========================================================
==*/

   /* (4) initier la négociation SSL via la connexion TCP */
/*=========================================================
==*/

   ssl = SSL_new(ctx); /* créer un point de terminaison de pile SSL */
SSL_set_fd(ssl, sd); /* attacher la pile SSL au socket */ err =
SSL_connect(ssl); /* lancer la négociation SSL */ printf("(4) Point de
terminaison SSL créé et négociation terminée\n\n");

   /*=========================================*/
```

```
   /* (5) affiche le chiffrement négocié choisi */
/*=========================================*/ printf("(5)
SSL connecté avec le chiffrement : %s\n\n", SSL_get_cipher(ssl));

   /*=========================================*/

   /* (6) imprimer le certificat du serveur */
/*=========================================*/ server_cert =
SSL_get_peer_certificate(ssl); printf("(6) le certificat du serveur a été
reçu :\n\n");

   str = X509_NAME_oneline(X509_get_subject_name(server_cert), 0,
0); printf(" sujet : %s\n", str);

   str = X509_NAME_oneline(X509_get_issuer_name(server_cert), 0,
0); printf(" issuer: %s\n\n", str); /* la vérification du certificat aurait lieu
ici */

   X509_free(certificat_serveur);


/*******************************************************
***/

   /* (7) poignée de main terminée --- envoyer une requête HTTP via
SSL */
/*******************************************************
/

   sprintf(host_header,"Hôte : %s:443\r\n",nom d'hôte);
strcpy(outbuf,"GET / HTTP/1.0\r\n"); strcat(outbuf,host_header);
strcat(outbuf,"Connexion : fermer\r\n"); strcat(outbuf,"\r\n");

   err = SSL_write(ssl, outbuf, strlen(outbuf)); shutdown (sd, 1); /*
envoyer EOF au serveur */ printf("(7) a envoyé une requête HTTP sur
un canal crypté :\n\n%s\n",outbuf);

   /***********************************************/

   /* (8) relire la réponse HTTP de la pile SSL */
/***********************************************/

   err = SSL_read(ssl, inbuf, sizeof(inbuf) - 1); inbuf[err] = '\0';

   printf ("(8) a récupéré %d octets de réponse
HTTP :\n\n%s\n",err,inbuf);

   /***********************************************/

   /* (9) tout est fait, donc fermez la connexion et nettoyez */

   /***********************************************/

   SSL_shutdown(ssl); fermer (sd);

   SSL_gratuit (ssl);
```

```
    SSL_CTX_free (ctx);

    printf("(9) tout est fait, nettoyé et connexion fermée\n\n"); }
```

Cet exemple est compilé et exécuté sous Sun Solaris, mais il illustre le fonctionnement des programmes SSL sur de nombreuses plateformes d'exploitation. L'ensemble du programme, y compris le chiffrement, la gestion des clés et des certificats, tient dans un programme C de trois pages, grâce aux puissantes fonctionnalités d'OpenSSL.

Parcourons le programme section par section :

• La partie supérieure du programme comprend les fichiers de support nécessaires à la prise en charge du réseau TCP et SSL.

• La section 1 crée le contexte local qui garde la trace des paramètres de poignée de main et d'autres états de la connexion SSL, en appelant SSL_CTX_new.

• La section 2 convertit le nom d'hôte d'entrée (fourni comme argument de ligne de commande) en adresse IP, à l'aide de la fonction Unix gethostbyname. D'autres plateformes peuvent proposer cette fonctionnalité d'autres manières.

• La section 3 ouvre une connexion TCP au port 443 sur le serveur en créant un socket local, en configurant les informations d'adresse distante et en se connectant au serveur distant.

• Une fois la connexion TCP établie, nous y attachons la couche SSL à l'aide de SSL_new et SSL_set_fd, puis nous établissons la liaison SSL avec le serveur en appelant SSL_connect. Une fois la section 4 terminée, nous disposons d'une connexion fonctionnelle.

Canal SSL établi, avec des chiffrements choisis et des certificats échangés.

• La section 5 imprime la valeur du chiffrement en masse choisi.

• La section 6 affiche certaines informations contenues dans les certificats X.509 renvoyés par le serveur, notamment sur le titulaire et l'organisation qui l'a émis. La bibliothèque OpenSSL n'utilise pas les informations du certificat du serveur. Une application SSL réelle, comme un navigateur web, effectuerait des vérifications sur le certificat pour s'assurer qu'il est correctement signé et provient du bon hôte. Nous avons discuté de ce que

les navigateurs utilisent les certificats de serveur dans « Validation du certificat de site ».

• À ce stade, notre connexion SSL est prête à être utilisée pour un transfert de données sécurisé. Dans la section 7, nous envoyons la simple requête HTTP « GET / HTTP/1.0 » sur le canal SSL à l'aide de SSL_write, puis fermons la partie sortante de la connexion.

• Dans la section 8, nous lisons la réponse de la connexion via SSL_read et l'affichons à l'écran. La couche SSL prenant en charge le chiffrement et le déchiffrement, nous pouvons simplement écrire et lire des commandes HTTP classiques.

• Enfin, nous faisons le ménage dans la section 9.

Consultez http://www.openssl.org pour plus d'informations sur les bibliothèques OpenSSL.

Exécution de notre client OpenSSL simple

L'exemple suivant illustre la sortie de notre client HTTP simple pointé vers un serveur sécurisé. Dans ce cas, nous avons redirigé le client vers la page d'accueil de la société de courtage en ligne Morgan Stanley. Les sociétés de trading en ligne utilisent largement le protocole HTTPS.

% https_client clients1.online.msdw.com

(1) Contexte SSL initialisé

(2) 'clients1.online.msdw.com' a l'adresse IP '63.151.15.11'

(3) Connexion TCP ouverte sur l'hôte « clients1.online.msdw.com », port 443

(4) Point de terminaison SSL créé et négociation terminée

(5) SSL connecté avec le chiffrement : DES-CBC3-MD5

(6) Le certificat du serveur a été reçu :

objet : /C=US/ST=Utah/L=Salt Lake City/O=Morgan Stanley/OU=Online/CN= clients1.online.msdw.com émetteur : /C=US/O=RSA Data Security, Inc./OU=Secure Server Certification Authority (7) a envoyé une requête HTTP via un canal chiffré :

OBTENIR / HTTP/1.0

Hébergeur : clients1.online.msdw.com:443

Connexion : close (8) a reçu 615 octets de réponse HTTP :

HTTP/1.1 302 trouvé

Date : sam. 9 mars 2002 09:43:42 GMT

Serveur : Stronghold/3.0 Apache/1.3.14 RedHat/3013c (Unix) mod_ssl/2.7.1 OpenSSL/0.9.6

Emplacement : https://clients.online.msdw.com/cgi-bin/ICenter/home

Connexion : fermer

Type de contenu : texte/html ; jeu de caractères = iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">

<HTML><HEAD>

<TITLE>302 Trouvé</TITLE>

</HEAD><BODY>

<H1>Trouvé</H1>

Le document a été déplacé <A HREF="https://clients.online.msdw.com/cgi-bin/ICenter/home">ici</A>.<P>

<HR>

<ADRESSE>Serveur Stronghold/3.0 Apache/1.3.14 RedHat/3013c sur clients1.online.msdw.com

Port 443</ADRESSE>

</BODY></HTML>

(9) tout est fait, nettoyé et connexion fermée

Une fois les quatre premières sections complétées, le client dispose d'une connexion SSL ouverte. Il peut alors s'informer de l'état de la connexion et des paramètres sélectionnés, et examiner les certificats du serveur.

Dans cet exemple, le client et le serveur ont négocié le chiffrement en masse DES-CBC3-MD5. Vous pouvez également constater que le certificat du site serveur appartient à l'organisation « Morgan Stanley » située à « Salt Lake City, Utah, États-Unis ». Le certificat a été délivré par RSA Data Security et le nom d'hôte est « clients1.online.msdw.com », ce qui correspond à notre requête.

Une fois le canal SSL établi et le client satisfait du certificat du site, il envoie sa requête HTTP via le canal sécurisé. Dans notre exemple, le client envoie une simple requête HTTP « GET / HTTP/1.0 » et reçoit une réponse de redirection 302, demandant à l'utilisateur de récupérer une autre URL.

Tunnellisation du trafic sécurisé via des proxys

Les clients utilisent souvent des serveurs proxy web pour accéder aux serveurs web en leur nom (les proxys sont abordés au chapitre 6). Par exemple, de nombreuses entreprises placent un proxy au périmètre de sécurité de leur réseau et de l'Internet public (figure 14-19). Le proxy est le seul périphérique autorisé par les routeurs pare-feu à échanger du trafic HTTP, et il peut utiliser des fonctions de détection de virus ou d'autres contrôles de contenu.

Figure 14-19. Proxy de pare-feu d'entreprise

Mais une fois que le client commence à chiffrer les données vers le serveur à l'aide de la clé publique de ce dernier, le proxy n'est plus en mesure de lire l'en-tête HTTP ! Et s'il ne peut pas le lire, il ne saura pas où transférer la requête (Figure 14-20).

Figure 14-20. Le proxy ne peut pas transmettre une requête chiffrée.

Pour que HTTPS fonctionne avec les proxys, quelques modifications sont nécessaires pour indiquer au proxy où se connecter. Une technique courante est le protocole de tunneling SSL HTTPS.

Tunnellisation du trafic sécurisé via des proxys

Grâce au protocole de tunneling HTTPS, le client indique d'abord au proxy l'hôte et le port sécurisés auxquels il souhaite se connecter. Il le fait en clair, avant le début du chiffrement, afin que le proxy puisse lire ces informations.

HTTP est utilisé pour envoyer les informations de point de terminaison en texte clair, grâce à une nouvelle méthode d'extension appelée CONNECT. Cette méthode indique au proxy d'établir une connexion avec l'hôte et le numéro de port souhaités, puis de tunneliser les données directement entre le client et le serveur. La méthode CONNECT est une commande d'une ligne qui fournit le nom d'hôte et le port du serveur d'origine sécurisé, séparés par deux points. L'expression hôte:port est suivie d'un espace et d'une chaîne de version HTTP, suivie d'un CRLF. Viennent ensuite une série de zéro, une ou plusieurs lignes d'en-tête de requête HTTP, suivies d'une ligne vide. Après cette ligne vide, si la connexion a réussi, le transfert de données SSL peut commencer. Voici un exemple :

CONNECTER home.netscape.com:443 HTTP/1.0

Agent utilisateur : Mozilla/1.1N

<les données brutes cryptées SSL suivraient ici...>

Après la ligne vide de la requête, le client attend une réponse du proxy. Ce dernier évalue la requête et s'assure qu'elle est valide et que l'utilisateur est autorisé à demander une telle connexion. Si tout est en ordre, le proxy établit une connexion au serveur de destination et, en cas de succès, envoie une réponse 200 « Connexion établie » au client.

HTTP/1.0 200 Connexion établie

Agent proxy : Netscape-Proxy/1.1

Pour plus d’informations sur les tunnels sécurisés et les proxys de sécurité, reportez-vous à « Tunnels » au chapitre 8.

Pour plus d'informations

La sécurité et la cryptographie sont des sujets extrêmement importants et complexes. Si vous souhaitez en savoir plus sur la sécurité HTTP, la cryptographie numérique, les certificats numériques et l'infrastructure à clés publiques, voici quelques points de départ.

Sécurité HTTP

Sécurité Web, confidentialité et commerce

Simson Garfinkel, O'Reilly & Associates, Inc. Il s'agit de l'une des meilleures et des plus lisibles introductions à la sécurité Web et à l'utilisation de SSL/TLS et des certificats numériques.

http://www.ietf.org/rfc/rfc2818.txt

La RFC 2818, « HTTP sur TLS », spécifie comment mettre en œuvre un protocole HTTP sécurisé sur

Transport Layer Security (TLS), le successeur moderne de SSL.
http://www.ietf.org/rfc/rfc2817.txt

La RFC 2817, « Mise à niveau vers TLS dans HTTP/1.1 », explique comment utiliser le mécanisme de mise à niveau de HTTP/1.1 pour initier TLS sur une connexion TCP existante. Cela permet au trafic HTTP sécurisé et non sécurisé de partager le même port connu (ici, http: à 80 plutôt que https: à 443). Cela permet également l'hébergement virtuel, de sorte qu'un seul serveur HTTP+TLS peut lever l'ambiguïté du trafic destiné à plusieurs noms d'hôtes sur une même adresse IP.

SSL et TLS

http://www.ietf.org/rfc/rfc2246.txt

La RFC 2246, « Le protocole TLS version 1.0 », spécifie la version 1.0 du protocole TLS (successeur de SSL). TLS assure la confidentialité des communications sur Internet. Ce protocole permet aux applications client/serveur de communiquer de manière à empêcher les écoutes clandestines, les falsifications et la falsification de messages.

http://developer.netscape.com/docs/manuals/security/sslin/contents.htm

« Introduction à SSL » présente le protocole Secure Sockets Layer (SSL). Initialement développé par Netscape, SSL est universellement accepté sur le Web pour les communications authentifiées et chiffrées entre clients et serveurs.

http://www.netscape.com/eng/ssl3/draft302.txt

« Le protocole SSL version 3.0 » est la spécification de Netscape pour SSL de 1996.

http://developer.netscape.com/tech/security/ssl/howitworks.html

« Comment fonctionne SSL » est l'introduction de Netscape à la cryptographie à clé.

http://www.openssl.org

Le projet OpenSSL est un effort collaboratif visant à développer une boîte à outils robuste, commerciale, complète et open source, implémentant les protocoles Secure Sockets Layer (SSL v2/v3) et Transport Layer Security (TLS v1), ainsi qu'une bibliothèque de cryptographie polyvalente et performante. Le projet est géré par une communauté mondiale de bénévoles qui utilisent Internet pour communiquer, planifier et développer la boîte à outils OpenSSL et sa documentation. OpenSSL s'appuie sur l'excellente bibliothèque SSLeay développée par Eric A. Young et Tim J. Hudson. La boîte à outils OpenSSL est sous licence de type Apache, ce qui signifie que vous êtes libre de l'obtenir et de l'utiliser à des fins commerciales et non commerciales, sous réserve de quelques conditions de licence simples.

Infrastructure à clé publique

http://www.ietf.org/html.charters/pkix-charter.html

Le groupe de travail IETF PKIX a été créé en 1995 dans le but de développer les normes Internet nécessaires pour prendre en charge une clé publique basée sur X.509.

Infrastructures. Voici un bon résumé des activités de ce groupe.

Pour plus d'informations

http://www.ietf.org/rfc/rfc2459.txt

La RFC 2459, « Certificat d'infrastructure à clé publique Internet X.509 et profil CRL », contient des détails sur les certificats numériques X.509 v3.

Cryptographie numérique

Cryptographie appliquée

Bruce Schneier, John Wiley & Sons. Un ouvrage classique sur la cryptographie pour les développeurs.

The Code Book : La science du secret, de l'Égypte ancienne à la cryptographie quantique. Simon Singh, Anchor Books. Ce livre divertissant est une introduction à la cryptographie. Bien qu'il ne soit pas destiné aux experts en technologie, il offre une exploration historique et vivante du codage secret.

PARTIE IV

IV. Entités, codages et

Internationalisation

La partie IV concerne les corps d'entité des messages HTTP et le contenu que les corps d'entité transportent sous forme de fret :

• Le chapitre 15, Entités et codages, décrit les formats et la syntaxe du contenu HTTP.

• Le chapitre 16, Internationalisation, examine les normes Web qui permettent aux utilisateurs d'échanger du contenu dans différentes langues et différents jeux de caractères, à travers le

globe.

• Le chapitre 17, Négociation de contenu et transcodage, explique les mécanismes de négociation de contenu acceptable.

Chapitre 15Ceci est le titre du livre CHAPITRE 15

Entités et codages

HTTP transporte chaque jour des milliards d'objets multimédias de toutes sortes. Images, textes, films, logiciels… tout est pris en charge par HTTP. HTTP veille également à ce que ses messages soient correctement transportés, identifiés, extraits et traités. Plus précisément, HTTP garantit que sa cargaison :

• Peut être identifié correctement (en utilisant les formats multimédias Content-Type et Content-

En-têtes de langue) afin que les navigateurs et autres clients puissent traiter correctement le contenu

• Peut être décompressé correctement (en utilisant les en-têtes Content-Length et Content-Encoding)

• Est frais (en utilisant des validateurs d'entité et des contrôles d'expiration du cache)

• Répond aux besoins de l'utilisateur (basé sur les en-têtes Accept de négociation de contenu)

• Se déplace rapidement et efficacement sur le réseau (en utilisant des requêtes de plage, le codage delta et d'autres compressions de données)

• Arrive complet et non falsifié (en utilisant les en-têtes de codage de transfert et les sommes de contrôle Content-MD5)

Pour que tout cela se produise, HTTP utilise des entités bien étiquetées pour transporter le contenu.

Ce chapitre aborde les entités, leurs en-têtes associés et leur fonctionnement pour le transport de données web. Nous montrerons comment HTTP fournit les informations essentielles sur la taille, le type et l'encodage du contenu. Nous expliquerons également certaines des fonctionnalités les plus complexes et les plus puissantes des entités HTTP, notamment les requêtes de plage, l'encodage delta, les résumés et

encodages fragmentés.

Ce chapitre couvre :

• Le format et le comportement des entités de message HTTP en tant que conteneurs de données HTTP

• Comment HTTP décrit la taille des corps d'entité et ce que HTTP exige en matière de dimensionnement

• Les en-têtes d'entité utilisés pour décrire le format, l'alphabet et la langue du contenu, afin que les clients puissent le traiter correctement

• Encodages de contenu réversibles, utilisés par les expéditeurs pour transformer le format des données de contenu avant l'envoi afin qu'il prenne moins de place ou soit plus sécurisé

• Codage de transfert, qui modifie la façon dont HTTP expédie les données pour améliorer la communication de certains types de contenu, et codage fragmenté, un codage de transfert qui découpe les données en plusieurs morceaux pour fournir en toute sécurité du contenu de longueur inconnue

• L'assortiment de balises, d'étiquettes, d'heures et de sommes de contrôle qui aident les clients à obtenir la dernière version du contenu demandé

• Les validateurs qui agissent comme des numéros de version sur le contenu, afin que les applications Web puissent s'assurer qu'elles ont un contenu récent, et les champs d'en-tête HTTP conçus pour contrôler la fraîcheur des objets

• Les plages, qui sont utiles pour reprendre les téléchargements interrompus là où ils se sont arrêtés

• Extensions d'encodage delta HTTP, qui permettent aux clients de demander uniquement les parties d'une page Web qui ont réellement changé depuis une révision précédemment consultée

• Sommes de contrôle des corps d'entité, qui sont utilisées pour détecter les changements dans le contenu de l'entité lorsqu'il passe par des proxys

Les messages sont des caisses, les entités sont des marchandises

Si l'on considère les messages HTTP comme les caisses du système de transport Internet, les entités HTTP constituent alors leur véritable cargaison. La figure 15-1 illustre une entité simple, contenue dans un message de réponse HTTP.

HTTP/1.0 200 OK

Serveur : Netscape-Enterprise/3.6

Date : dimanche 17 septembre 2000 00:01:05 GMT

Type de contenu : texte/brut

Longueur du contenu : 18 Salut ! Je suis un message ! En-têtes d'entité

Corps d'entité

Entité

Figure 15-1. L'entité message est composée d'en-têtes et d'un corps d'entité.

Les en-têtes d'entité indiquent un document en texte brut (Content-Type : text/plain) de seulement 18 caractères (Content-Length : 18). Comme toujours, une ligne vide (CRLF) sépare les champs d'en-tête du début du corps du document.

Les en-têtes d'entité HTTP (abordés au chapitre 3) décrivent le contenu d'un message HTTP. HTTP/1.1 définit dix champs d'en-tête d'entité principaux :

Type de contenu

Le type d'objet transporté par l'entité.

Longueur du contenu

La longueur ou la taille du message envoyé.

Contenu-Langue

Le langage humain qui correspond le mieux à l’objet envoyé.

Codage de contenu

Toute transformation (compression, etc.) effectuée sur les données de l'objet.

Contenu-Emplacement

Un emplacement alternatif pour l'objet au moment de la demande.

Gamme de contenu

S'il s'agit d'une entité partielle, cet en-tête définit quels éléments de l'ensemble sont inclus.

Contenu-MD5

Une somme de contrôle du contenu du corps de l'entité.

Dernière modification

La date à laquelle ce contenu particulier a été créé ou modifié sur le serveur.

Expire

La date et l'heure à laquelle ces données d'entité deviendront obsolètes.

Permettre

Quelles méthodes de requête sont légales sur cette ressource ; par exemple, GET et HEAD.

ETag

Un validateur unique pour cette instance* particulière du document. L'en-tête ETag n'est pas formellement défini comme un en-tête d'entité, mais il est important pour de nombreuses opérations impliquant des entités.

Contrôle du cache

Directives sur la mise en cache de ce document. L'en-tête Cache-Control, comme l'en-tête ETag, n'est pas formellement défini comme un en-tête d'entité.

Organismes d'entité

Le corps de l'entité contient uniquement les données brutes.† Toute autre information descriptive est contenue dans les en-têtes. Comme le corps de l'entité ne contient que des données brutes, les en-têtes d'entité sont nécessaires pour en décrire la signification. Par exemple, l'en-tête d'entité ContentType indique comment interpréter les données (image, texte, etc.), et l'en-tête d'entité Content-Encoding indique si les données ont été compressées ou recodées. Nous aborderons tout cela et bien plus encore dans les sections suivantes.

Le contenu brut commence immédiatement après la ligne CRLF vide qui marque la fin des champs d'en-tête. Quel que soit le contenu (texte ou binaire, document ou image, compressé ou non, anglais, français ou japonais), il est placé juste après la ligne CRLF.

* Les instances sont décrites plus loin dans ce chapitre, dans la section « Instances variables dans le temps ».

† S'il existe un en-tête Content-Encoding, le contenu a déjà été codé par l'algorithme de codage de contenu et le premier octet de l'entité est le premier octet de la cargaison codée (par exemple, compressée).

Les messages sont des caisses, les entités sont des marchandises

La figure 15-2 présente deux exemples de messages HTTP réels, l'un contenant du texte, l'autre une image. Les valeurs hexadécimales indiquent le contenu exact du message :

• Dans la figure 15-2a, le corps de l'entité commence à l'octet 65, juste après le CRLF de fin d'en-tête. Il contient les caractères ASCII pour « Salut ! Je suis un message ! »

• Dans la figure 15-2b, le corps de l'entité commence à l'octet 67. Il contient le contenu binaire de l'image GIF. Les fichiers GIF commencent par une signature de version de 6 octets, une largeur et une hauteur de 16 bits. Ces trois éléments sont visibles directement dans le corps de l'entité.

Figure 15-2. Vidages hexadécimaux du contenu réel du message (le contenu brut du message suit un CRLF vide)

Longueur du contenu : la taille de l'entité

L'en-tête Content-Length indique la taille du corps de l'entité dans le message, en octets. Cette taille inclut tous les encodages de contenu (la taille Content-Length d'un fichier texte compressé au format gzip correspond à la taille compressée, et non à la taille d'origine).

L'en-tête Content-Length est obligatoire pour les messages avec corps d'entité, sauf si le message est transporté par encodage fragmenté. Content-Length est nécessaire pour détecter une troncature prématurée des messages en cas de panne du serveur et pour segmenter correctement les messages partageant une connexion persistante.

Détection de troncature

Les anciennes versions de HTTP utilisaient la méthode de fermeture de connexion pour délimiter la fin d'un message. Cependant, sans Content-Length, les clients ne peuvent pas faire la distinction entre une fermeture de connexion réussie à la fin d'un message et une fermeture due à une panne du serveur en cours de message. Les clients ont besoin de Content-Length pour détecter la troncature des messages.

La troncature des messages est particulièrement grave pour les serveurs proxy de mise en cache. Si un cache reçoit un message tronqué et ne reconnaît pas la troncature, il peut stocker le contenu défectueux et le diffuser plusieurs fois. Les serveurs proxy de mise en cache ne mettent généralement pas en cache les corps HTTP sans en-tête Content-Length explicite, afin de réduire le risque de mise en cache de messages tronqués.

Longueur de contenu incorrecte

Une longueur de contenu incorrecte peut causer encore plus de dommages qu'un contenu manquant.

Longueur. Certains clients et serveurs anciens présentaient des bugs connus concernant le calcul de la longueur du contenu. C'est pourquoi certains clients, serveurs et proxys utilisent des algorithmes pour détecter et corriger les interactions avec les serveurs défectueux. Les agents utilisateurs HTTP/1.1 sont officiellement censés avertir l'utilisateur lorsqu'une longueur non valide est reçue et détectée.

Longueur du contenu et connexions persistantes

La longueur du contenu est essentielle pour les connexions persistantes. Si la réponse rencontre une connexion persistante, une autre réponse HTTP peut immédiatement suivre la réponse actuelle. L'en-tête « Content-Length » indique au client où se termine un message et où commence le suivant. La connexion étant persistante, le client ne peut pas utiliser « connection close » pour identifier la fin du message. Sans en-tête « Content-Length », les applications HTTP ne sauront pas où se termine un corps d'entité et où commence le message suivant.

Comme nous le verrons dans « Codage de transfert et codage segmenté », il existe une situation où vous pouvez utiliser des connexions persistantes sans en-tête Content-Length : l'encodage segmenté. Ce codage envoie les données en une série de segments, chacun ayant une taille spécifique. Même si le serveur ne connaît pas la taille de l'entité entière au moment de la génération des en-têtes (souvent parce que l'entité est générée dynamiquement), il peut utiliser l'encodage segmenté pour transmettre des segments de taille bien définie.

Codage du contenu

HTTP permet d'encoder le contenu d'un corps d'entité, par exemple pour le sécuriser ou le compresser afin de gagner de la place (la compression est expliquée en détail plus loin dans ce chapitre). Si le corps a été encodé, l'en-tête Content-Length spécifie la longueur, en octets, du corps encodé, et non celle du corps d'origine non encodé.

Certaines applications HTTP sont connues pour commettre des erreurs et envoyer la taille des données avant l'encodage, ce qui entraîne de graves erreurs, notamment avec les connexions persistantes. Malheureusement, aucun des en-têtes décrits dans HTTP/1.1 n'est pris en charge.

Longueur du contenu : la taille de l'entité

la spécification peut être utilisée pour envoyer la longueur du corps original non codé, qui

rend difficile pour les clients de vérifier l'intégrité de leurs processus de décryptage.*

Règles de détermination de la longueur du corps d'une entité

Les règles suivantes décrivent comment déterminer correctement la longueur et la fin d'un corps d'entité dans différentes circonstances. Les règles doivent être appliquées dans l'ordre ; la première correspondance s'applique.

1. Si un type de message HTTP particulier n'est pas autorisé à contenir un corps, ignorez l'en-tête Content-Length pour le calcul du corps. Dans ce cas, les en-têtes Content-Length sont informatifs et ne décrivent pas la longueur réelle du corps. (Les applications HTTP naïves peuvent être en difficulté si elles supposent que Content-Length implique systématiquement la présence d'un corps.)

L'exemple le plus significatif est la réponse HEAD. La méthode HEAD demande au serveur d'envoyer les en-têtes qui auraient été renvoyés par une requête GET équivalente, mais sans corps. Comme une réponse GET renvoie un en-tête Content-Length, la réponse HEAD en renvoie également un. Cependant, contrairement à la réponse GET, elle n'a pas de corps. Les réponses 1XX, 204 et 304 peuvent également contenir des en-têtes Content-Length informatifs, mais sans corps d'entité. Les messages interdisant les corps d'entité doivent se terminer à la première ligne vide après les en-têtes, quels que soient les champs d'en-tête d'entité présents.

2. Si un message contient un en-tête Transfer-Encoding (autre que l'encodage HTTP « identity » par défaut), l'entité sera terminée par un modèle spécial appelé « bloc de zéro octet », à moins que le message ne soit d'abord terminé par la fermeture de la connexion. Nous aborderons les encodages de transfert et les encodages fragmentés plus loin dans ce chapitre.

3. Si un message comporte un en-tête Content-Length (et que le type de message autorise les corps d'entité), la valeur Content-Length contient la longueur du corps, sauf s'il contient un en-tête Transfer-Encoding non identitaire. Si un message est reçu avec un champ d'en-tête Content-Length et un champ d'en-tête Transfer-Encoding non identitaire, vous devez ignorer Content-Length, car l'encodage de transfert modifiera la représentation et le transfert des corps d'entité (et probablement le nombre d'octets transmis).

4. Si le message utilise le type de média « multipart/byteranges » et que la longueur de l'entité n'est pas spécifiée (dans l'en-tête Content-Length), chaque partie du message multipart aura sa propre taille. Ce type multipart est le seul type de corps d'entité à auto-délimiter sa taille ; il ne doit donc être envoyé que si l'expéditeur sait que le destinataire peut l'analyser.†

* L'en-tête Content-MD5, qui permet d'envoyer le MD5 128 bits du document, contient également le MD5 du document encodé. Cet en-tête est décrit plus loin dans ce chapitre.

† Étant donné qu'un en-tête Range peut être transmis par un proxy plus primitif qui ne comprend pas les plages multiparties/octets, l'expéditeur doit délimiter le message à l'aide des méthodes 1, 3 ou 5 de cette section s'il n'est pas sûr que le récepteur comprenne le format d'auto-délimitation.

5. Si aucune des règles ci-dessus ne correspond, l'entité se termine à la fermeture de la connexion. En pratique, seuls les serveurs peuvent utiliser la fermeture de connexion pour signaler la fin d'un message. Les clients ne peuvent pas fermer la connexion pour signaler la fin de leurs messages, car cela empêcherait le serveur de renvoyer une réponse.*

6. Pour être compatible avec les applications HTTP/1.0, toute requête HTTP/1.1 comportant un corps d'entité doit également inclure un champ d'en-tête Content-Length valide (sauf si le serveur est reconnu comme étant conforme à HTTP/1.1). La spécification HTTP/1.1 recommande que si une requête contient un corps mais pas de champ Content-Length, le serveur envoie une réponse 400 « Requête incorrecte » s'il ne peut déterminer la longueur du message, ou une réponse 411 « Longueur requise » s'il souhaite obtenir un champ Content-Length valide.

Résumés d'entités

Bien que HTTP soit généralement implémenté via un protocole de transport fiable tel que TCP/IP, des parties de messages peuvent être modifiées en transit pour diverses raisons, telles que des proxys de transcodage non conformes ou des proxys intermédiaires bogués. Pour détecter toute modification involontaire (ou indésirable) des données du corps de l'entité, l'expéditeur peut générer une somme de contrôle des données lors de la génération de l'entité initiale, et le destinataire peut vérifier l'intégrité de cette somme de contrôle afin de détecter toute modification involontaire de l'entité.†

L'en-tête Content-MD5 est utilisé par les serveurs pour transmettre le résultat de l'exécution de l'algorithme MD5 sur le corps de l'entité. Seul le serveur d'origine de la réponse est autorisé à calculer et à transmettre l'en-tête Content-MD5. Les proxys et caches intermédiaires ne peuvent ni modifier ni ajouter l'en-tête, ce qui contreviendrait à l'objectif même de vérification de l'intégrité de bout en bout. L'en-tête Content-MD5 contient le MD5 du contenu après l'application de tous les codages de contenu au corps de l'entité et avant l'application de tout codage de transfert. Les clients souhaitant vérifier l'intégrité du message doivent d'abord décoder les codages de transfert, puis calculer le MD5 du corps de l'entité non codé résultant. Par exemple, si un document est compressé à l'aide de l'algorithme gzip, puis envoyé

avec un codage fragmenté, l'algorithme MD5 est exécuté sur l'intégralité du corps de l'entité.

Outre la vérification de l'intégrité des messages, l'en-tête MD5 peut servir de clé dans une table de hachage pour localiser rapidement des documents et réduire le stockage dupliqué de contenu. Malgré ces utilisations possibles, l'en-tête Content-MD5 est rarement envoyé.

* Le client pourrait effectuer une demi-fermeture de sa connexion de sortie uniquement, mais de nombreuses applications serveur ne sont pas conçues pour gérer cette situation et interpréteront une demi-fermeture comme une déconnexion du client. La gestion des connexions n'a jamais été clairement définie dans HTTP. Voir le chapitre 4 pour plus de détails.

† Cette méthode n'est bien sûr pas à l'abri d'une attaque malveillante qui remplacerait à la fois le corps du message et l'en-tête du résumé. Elle vise uniquement à détecter les modifications involontaires. D'autres fonctionnalités, telles que l'authentification par résumé, sont nécessaires pour se protéger contre toute altération malveillante.

Résumés d'entités

Les extensions HTTP ont proposé d'autres algorithmes de résumé dans les projets de l'IETF. Ces extensions ont proposé un nouvel en-tête, Want-Digest, qui permet aux clients de spécifier le type de résumé attendu avec la réponse. Les valeurs de qualité peuvent être utilisées pour suggérer plusieurs algorithmes de résumé et indiquer une préférence.

Type de média et jeu de caractères

Le champ d'en-tête Content-Type décrit le type MIME du corps de l'entité.* Le type MIME est un nom normalisé qui décrit le type de média sous-jacent transporté comme fret (fichier HTML, document Microsoft Word, vidéo MPEG, etc.). Les applications clientes utilisent le type MIME pour déchiffrer et traiter correctement le contenu.

Les valeurs Content-Type sont des types MIME standardisés, enregistrés auprès de l'IANA (Internet Assigned Numbers Authority). Les types MIME se composent d'un type de média principal (par exemple, texte, image, audio), suivi d'une barre oblique, puis d'un sous-type qui précise le type de média. Le tableau 15-1 répertorie quelques types MIME courants pour l'en-tête ContentType. D'autres types MIME sont répertoriés dans l'annexe D.

Tableau 15-1. Types de supports courants

Type de média Description

Le corps de l'entité text/html est un document HTML

texte/brut Le corps de l'entité est un document en texte brut

image/gif Le corps de l'entité est une image de type GIF

image/jpeg Le corps de l'entité est une image de type JPEG

Le corps de l'entité audio/x-wav contient des données sonores WAV

Le corps de l'entité modèle/vrml est un modèle VRML tridimensionnel

application/vnd.ms-powerpoint Le corps de l'entité est une présentation Microsoft PowerPoint

multipart/byteranges Le corps de l'entité comporte plusieurs parties, chacune contenant une plage différente (en octets) du document complet

message/http Le corps de l'entité contient un message HTTP complet (voir TRACE)

Il est important de noter que l'en-tête Content-Type spécifie le type de média du corps de l'entité d'origine. Si l'entité a subi un codage de contenu, par exemple, l'en-tête Content-Type spécifiera toujours le type de corps de l'entité avant le codage.

* Dans le cas de la requête HEAD, Content-Type affiche le type qui aurait été envoyé s'il s'agissait d'une requête GET.

Codages de caractères pour les médias textuels

L'en-tête Content-Type prend également en charge des paramètres facultatifs permettant de préciser le type de contenu. Le paramètre « charset » en est le principal exemple : il spécifie le mécanisme de conversion des bits de l'entité en caractères dans un fichier texte.

Type de contenu : texte/html ; jeu de caractères = iso-8859-4

Nous parlons des jeux de caractères en détail au chapitre 16.

Types de supports en plusieurs parties

Les messages électroniques MIME « multiparties » contiennent plusieurs messages regroupés et envoyés comme un seul message complexe. Chaque composant est autonome, avec son propre ensemble d'en-têtes décrivant son contenu ; les différents composants sont concaténés et délimités par une chaîne.

HTTP prend également en charge les corps en plusieurs parties ; cependant, ils ne sont généralement envoyés que dans l'une des deux situations suivantes : dans les soumissions de formulaires à remplir et dans les réponses de plage contenant des parties d'un

document.

Soumissions de formulaires en plusieurs parties

Lors de la soumission d'un formulaire HTTP, les champs de texte de longueur variable et les objets téléchargés sont envoyés séparément

dans un corps en plusieurs parties, permettant ainsi de remplir les formulaires avec des valeurs de types et de longueurs différents. Par exemple, vous pouvez choisir de remplir un formulaire demandant votre nom, une description avec votre pseudo et une petite photo, tandis qu'un ami peut indiquer son nom complet et un long essai décrivant sa passion pour la réparation des bus Volkswagen.

HTTP envoie de telles requêtes avec un en-tête Content-Type: multipart/form-data ou un en-tête Content-Type: multipart/mixed et un corps multipart, comme ceci :

Content-Type : multipart/form-data ; boundary=[abcdefghijklmnopqrstuvwxyz] où la limite spécifie la chaîne de délimitation entre les différentes parties du corps.

L'exemple suivant illustre l'encodage de données multipart/formulaire. Supposons que nous ayons ce formulaire :

<FORM action="http://server.com/cgi/handle" enctype="multipart/form-data" method="post">

<P>

Quel est votre nom ? <INPUT type="text" name="submit-name">

Quels fichiers envoyez-vous ? <INPUT type="file" name="files">

<INPUT type="submit" value="Envoyer"> <INPUT type="reset">

</FORM>

Type de média et jeu de caractères

Si l'utilisateur saisit « Sally » dans le champ de saisie de texte et sélectionne le fichier texte « essayfile.txt », l'agent utilisateur peut renvoyer les données suivantes :

Type de contenu : multipart/form-data ; boundary=AaB03x

--AaB03x

Contenu-Disposition : données-formulaires ; nom = nom-de-soumission »

Sortie

--AaB03x

Contenu-Disposition : données de formulaire ; nom = fichiers ; nom de fichier = essayfile.txt ;

Type de contenu : texte/brut ...contenu de essayfile.txt...

--AaB03x--

Si l'utilisateur sélectionne un deuxième fichier (image), « imagefile.gif », l'agent utilisateur peut construire les parties comme suit :

Type de contenu : multipart/form-data ; boundary=AaB03x

--AaB03x

Contenu-Disposition : données-formulaires ; nom = nom-de-soumission »

Sortie

--AaB03x

Contenu-Disposition : données de formulaire ; nom = « fichiers »

Type de contenu : multipart/mixte ; boundary=BbC04y

--BbC04y

Contenu-Disposition : fichier ; nom de fichier ="essayfile.txt"

Type de contenu : texte/brut ...contenu de essayfile.txt...

--BbC04y

Contenu-Disposition : fichier ; nom de fichier = imagefile.gif »

Type de contenu : image/gif

Content-Transfer-Encoding : binaire...contenu de imagefile.gif...

--BbC04y--

--AaB03x--

Réponses à plusieurs parties

Les réponses HTTP aux requêtes de plage peuvent également être multiparties. Elles contiennent un en-tête Content-Type: multipart/byteranges et un corps multipartie contenant les différentes plages. Voici un exemple de réponse multipartie à une requête portant sur différentes plages d'un document :

HTTP/1.0 206 Contenu partiel

Serveur : Microsoft-IIS/5.0

Date : dim. 10 déc. 2000 19:11:20 GMT

Emplacement du contenu : http://www.joes-hardware.com/gettysburg.txt

Type de contenu : multipart/x-byteranges ; boundary=--[abcdefghijklmnopqrstuvwxyz]--

Dernière modification : sam. 9 déc. 2000 00:38:47 GMT

--[abcdefghijklmnopqrstuvwxyz]--

Type de contenu : texte/brut

Plage de contenu : octets 0-174/1 441

Il y a quatre-vingt-sept ans, nos pères ont donné naissance sur ce continent à une nouvelle nation, conçue dans la liberté et vouée à la proposition selon laquelle tous les hommes sont créés égaux.

--[abcdefghijklmnopqrstuvwxyz]--

Type de contenu : texte/brut

Plage de contenu : octets 552-761/1 441

Mais, plus largement, nous ne pouvons ni dédier, ni consacrer, ni sanctifier cette terre. Les hommes courageux, vivants et morts, qui ont lutté ici l'ont consacrée bien au-delà de notre faible pouvoir d'y ajouter ou d'en retrancher.

--[abcdefghijklmnopqrstuvwxyz]--

Type de contenu : texte/brut

Plage de contenu : octets 1 344-1 441/1 441

et que le gouvernement du peuple, par le peuple, pour le peuple ne disparaîtra pas de la terre.

--[abcdefghijklmnopqrstuvwxyz]--

Les demandes de portée sont abordées plus en détail plus loin dans ce chapitre.

Codage du contenu

Les applications HTTP souhaitent parfois encoder le contenu avant de l'envoyer. Par exemple, un serveur peut compresser un document HTML volumineux avant de l'envoyer à un client connecté via une connexion lente, afin de réduire le temps de transmission de l'entité. Un serveur peut également chiffrer le contenu afin d'empêcher toute consultation par des tiers non autorisés.

Ces types de codages sont appliqués au contenu de l'expéditeur. Une fois le contenu codé, les données codées sont envoyées au destinataire dans le corps de l'entité.

habituel.

Le processus de codage du contenu

Le processus d'encodage du contenu est :

1. Un serveur Web génère un message de réponse original, avec les en-têtes ContentType et Content-Length d'origine.

2. Un serveur de codage de contenu (peut-être le serveur d'origine ou un proxy en aval) crée un message codé. Ce message possède le même type de contenu, mais une longueur de contenu différente (si, par exemple, le corps est compressé). Le serveur de codage de contenu

ajoute un en-tête « Content-Encoding » au message codé afin qu'une application réceptrice puisse le décoder.

3. Un programme récepteur reçoit le message codé, le décode et obtient le

original.

Codage du contenu

La figure 15-3 illustre un exemple de codage de contenu.

Figure 15-3. Exemple de codage de contenu

Ici, une page HTML est encodée par une fonction d'encodage de contenu gzip, afin de produire un corps compressé plus petit. Ce corps compressé est envoyé sur le réseau, marqué avec l'encodage gzip. Le client destinataire décompresse l'entité à l'aide du décodeur gzip.

Cet extrait de réponse montre un autre exemple de réponse codée (une image compressée) :

HTTP/1.1 200 OK

Date : vendredi 5 novembre 1999 22:35:15 GMT

Serveur : Apache/1.2.4

Contenu-Longueur : 6096

Type de contenu : image/gif

Codage de contenu : gzip

[...]

Notez que l'en-tête Content-Type peut et doit rester présent dans le message. Il décrit le format d'origine de l'entité, informations qui peuvent être nécessaires à son affichage une fois décodée. N'oubliez pas que l'en-tête Content-Length représente désormais la longueur du corps encodé.

Types de codage de contenu

HTTP définit quelques types d'encodage de contenu standard et permet l'ajout d'encodages supplémentaires comme encodages d'extension. Les encodages sont normalisés par l'IANA, qui attribue un jeton unique à chaque algorithme d'encodage de contenu. L'en-tête Content-Encoding utilise ces valeurs de jeton standardisées pour décrire l'algorithme utilisé.

Certains des jetons de codage de contenu courants sont répertoriés dans le tableau 15-2.

Tableau 15-2. Jetons de codage de contenu

Valeur de codage de contenu Description

gzip Indique que l'encodage GNU zip a été appliqué à l'entité.a

compress Indique que le programme de compression de fichiers Unix a été exécuté sur l'entité.

deflate Indique que l'entité a été compressée au format zlib.b

identité : indique qu'aucun codage n'a été effectué sur l'entité. L'absence d'en-tête Content-Encoding peut être considérée comme une hypothèse.

a RFC 1952 décrit l'encodage gzip. b Les RFC 1950 et 1951 décrivent le format zlib et la compression deflate.

Les codages gzip, compress et deflate sont des algorithmes de compression sans perte utilisés pour réduire la taille des messages transmis sans perte d'informations. Parmi eux, gzip est généralement l'algorithme de compression le plus efficace et le plus utilisé.

En-têtes d'acceptation et de codage

Bien entendu, nous ne souhaitons pas que les serveurs encodent le contenu d'une manière que le client ne puisse pas déchiffrer. Pour empêcher les serveurs d'utiliser des encodages non pris en charge par le client, ce dernier transmet une liste des encodages de contenu pris en charge dans l'en-tête de requête Accept-Encoding. Si la requête HTTP ne contient pas d'en-tête Accept-Encoding, le serveur peut supposer que le client acceptera n'importe quel encodage (équivalent à la transmission de Accept-Encoding: *).

La figure 15-4 montre un exemple d’Accept-Encoding dans une transaction HTTP.

Figure 15-4. Codage du contenu

Codage du contenu

Le champ « Accept-Encoding » contient une liste d'encodages pris en charge, séparés par des virgules. Voici quelques exemples :

Accepter-Encodage : compresser, gzip Accepter-Encodage :

Accepter-Encodage : *

Accepter-Encodage : compresser ; q = 0,5, gzip ; q = 1,0

Accepter-Encodage : gzip ; q = 1,0, identité ; q = 0,5, * ; q = 0

Les clients peuvent indiquer leurs encodages préférés en associant des valeurs Q (qualité) à chaque encodage. Ces valeurs peuvent aller de 0,0, indiquant que le client ne souhaite pas l'encodage associé, à 1,0, indiquant l'encodage préféré. Le jeton « * » signifie « autre chose ». Le

choix de l'encodage de contenu à appliquer s'inscrit dans un processus plus général de choix du contenu à renvoyer au client dans une réponse. Ce processus, ainsi que les en-têtes Content-Encoding et Accept-Encoding, sont détaillés au chapitre 17.

Le jeton de codage d'identité ne peut être présent que dans l'en-tête Accept-Encoding et est utilisé par les clients pour spécifier une préférence relative par rapport aux autres algorithmes de codage de contenu.

Codage de transfert et codage par blocs

La section précédente a abordé les encodages de contenu, c'est-à-dire les transformations réversibles appliquées au corps du message. Les encodages de contenu sont étroitement liés aux détails du format de contenu concerné. Par exemple, vous pouvez compresser un fichier texte avec gzip, mais pas un fichier JPEG, car les JPEG ne se compressent pas bien avec gzip.

Cette section traite des codages de transfert. Ces derniers sont également des transformations réversibles appliquées au corps de l'entité, mais leur application est liée à l'architecture et est indépendante du format du contenu. L'application d'un codage de transfert à un message modifie le mode de transfert des données sur le réseau (Figure 15-5).

Transport sécuritaire

Historiquement, des codages de transfert existent dans d'autres protocoles pour assurer un « transport sécurisé » des messages sur un réseau. Le concept de transport sécurisé a une portée différente pour HTTP, où l'infrastructure de transport est standardisée et plus souple.

HTTP, le transport des corps de messages peut poser problème pour quelques raisons. Deux d'entre elles :

Taille inconnue

Certaines applications de passerelle et encodeurs de contenu ne peuvent pas déterminer la taille finale du corps d'un message sans générer le contenu au préalable. Souvent, ces serveurs préfèrent envoyer les données avant même d'en connaître la taille.

Réponse codée par le contenu

HTTP/1.0 200 OK

Codage du contenu : gzip

Type de contenu : texte/html

Bloc d'en-tête normal

[message codé]

[...]

Entité normale

(juste encodé)

Réponse codée par transfert

HTTP/1.1 200 OK

Codage de transfert : fragmenté

En-tête de base

Blocs codés

10 abcdefghijk

1 a Un message codé par contenu code uniquement la section entité du message. Avec les messages codés par transfert, le codage est une fonction du message entier, modifiant sa structure.

Figure 15-5. Codages de contenu et de transfert

HTTP nécessite que l'en-tête Content-Length précède les données, certains serveurs appliquent un codage de transfert pour envoyer les données avec un pied de page de fin spécial qui indique la fin des données.*

Sécurité

Vous pouvez utiliser un codage de transfert pour brouiller le contenu du message avant de l'envoyer sur un réseau de transport partagé. Cependant, en raison de la popularité des protocoles de sécurité de la couche transport comme SSL, la sécurité par codage de transfert est peu répandue.

En-têtes de codage de transfert

Il n'y a que deux en-têtes définis pour décrire et contrôler l'encodage du transfert :

Transfert-Encodage

Indique au récepteur quel codage a été effectué sur le message afin qu'il soit transporté en toute sécurité

TE

Utilisé dans l'en-tête de la requête pour indiquer au serveur quels codages de transfert d'extension peuvent être utilisés†

* Vous pourriez fermer la connexion en guise de signal de fin de message pour le « pauvre homme », mais cela interrompt les connexions persistantes.

† La signification de l'en-tête TE serait plus intuitive s'il était appelé en-tête Accept-Transfer-Encoding.

Codage de transfert et codage par blocs

Dans l'exemple suivant, la requête utilise l'en-tête TE pour indiquer au serveur qu'il accepte l'encodage fragmenté (ce qu'il doit faire s'il s'agit d'une application HTTP 1.1) et qu'il est prêt à accepter les bandes-annonces à la fin des messages encodés fragmentés :

OBTENIR /nouveaux_produits.html HTTP/1.1

Hébergeur : www.joes-hardware.com

Agent utilisateur : Mozilla/4.61 [fr] (WinNT ; I)

TE : bandes-annonces, en morceaux...

La réponse inclut un en-tête Transfer-Encoding pour indiquer au récepteur que le message a été codé par transfert avec le codage fragmenté :

HTTP/1.1 200 OK

Transfert-Encodage : fragmenté

Serveur : Apache/3.0 ...

Après cet en-tête initial, la structure du message va changer.

Toutes les valeurs de codage de transfert sont insensibles à la casse. HTTP/1.1 utilise des valeurs de codage de transfert dans les champs d'en-tête TE et Transfer-Encoding. La dernière spécification HTTP ne définit qu'un seul codage de transfert : le codage fragmenté.

L'en-tête TE, comme l'en-tête Accept-Encoding, peut comporter des valeurs Q pour décrire les formes préférées d'encodage de transfert. Cependant, la spécification HTTP/1.1 interdit l'association d'une valeur Q de 0,0 à l'encodage fragmenté.

Les futures extensions de HTTP pourraient nécessiter des encodages de transfert supplémentaires. Dans ce cas, l'encodage de transfert fragmenté doit toujours être appliqué par-dessus les encodages de transfert d'extension. Cela garantit que les données seront « tunnelisées » via les applications HTTP/1.1 qui comprennent l'encodage fragmenté, mais pas les autres encodages de transfert.

Encodage fragmenté

Le codage segmenté divise les messages en segments de taille connue. Chaque segment est envoyé l'un après l'autre, éliminant ainsi la nécessité de connaître la taille du message complet avant son envoi.

Notez que le codage fragmenté est une forme de codage de transfert et constitue donc un attribut du message, et non du corps. Le codage

multipartite, décrit précédemment dans ce chapitre, est un attribut du corps et est totalement distinct du codage fragmenté.

Connexions persistantes et en blocs

Lorsque la connexion entre le client et le serveur n’est pas persistante, les clients n’ont pas besoin de connaître la taille du corps qu’ils lisent : ils s’attendent à lire le corps jusqu’à ce que le serveur ferme la connexion.

Avec les connexions persistantes, la taille du corps du message doit être connue et envoyée dans l'en-tête Content-Length avant son écriture. Lorsque le contenu est créé dynamiquement sur un serveur, il peut être impossible de connaître la longueur du corps avant son envoi.

L'encodage segmenté offre une solution à ce problème : les serveurs peuvent envoyer le corps du message par segments, en spécifiant uniquement la taille de chaque segment. Le corps étant généré dynamiquement, le serveur peut en mettre une partie en mémoire tampon, envoyer sa taille et le segment, puis répéter le processus jusqu'à l'envoi complet du corps. Le serveur peut signaler la fin du corps avec un segment de taille nulle tout en maintenant la connexion ouverte et prête pour la réponse suivante.

Le codage segmenté est relativement simple. La figure 15-6 illustre l'anatomie de base d'un message segmenté. Il commence par un bloc d'en-tête de réponse HTTP initial, suivi d'un flux de segments. Chaque segment contient une valeur de longueur et ses données. La valeur de longueur est au format hexadécimal et est séparée des données du segment par un CRLF. La taille des données du segment est mesurée en octets et n'inclut ni la séquence CRLF entre la valeur de longueur et les données, ni la séquence CRLF à la fin du segment.

Le dernier morceau est spécial : il a une longueur de zéro, ce qui signifie « fin du corps ».

Figure 15-6. Anatomie d'un message fragmenté

Codage de transfert et codage par blocs

Un client peut également envoyer des données fragmentées à un serveur. Comme le client ne sait pas à l'avance si le serveur accepte le codage fragmenté (les serveurs n'envoient pas d'en-têtes TE dans leurs réponses), il doit se préparer à ce que le serveur rejette la requête fragmentée avec une réponse 411 « Longueur requise ».

Bandes-annonces dans des messages fragmentés

Une bande-annonce peut être ajoutée à un message fragmenté si l'en-tête TE du client indique qu'il accepte les bandes-annonces, ou si la

bande-annonce est ajoutée par le serveur qui a créé la réponse d'origine et que le contenu de la bande-annonce est une métadonnée facultative que le client n'a pas besoin de comprendre et d'utiliser (le client peut ignorer et supprimer le contenu de la bande-annonce).*

La fin du message peut contenir des champs d'en-tête supplémentaires dont les valeurs pourraient être inconnues au début du message (par exemple, parce que le contenu du corps du message a dû être généré au préalable). L'en-tête ContentMD5 est un exemple d'en-tête pouvant être inclus dans la fin du message : il serait difficile de calculer le MD5 d'un document avant sa génération. La figure 15-6 illustre l'utilisation des fins de message. Les en-têtes de message contiennent un en-tête « Frein » listant les en-têtes qui suivront le message fragmenté. Le dernier fragment est suivi des en-têtes listés dans l'en-tête « Frein ».

N'importe lequel des en-têtes HTTP peut être envoyé en tant que bandes-annonces, à l'exception des en-têtes Transfer-Encoding, Trailer et Content-Length.

Combinaison des codages de contenu et de transfert

L'encodage de contenu et l'encodage de transfert peuvent être utilisés simultanément. Par exemple, la figure 15-7 illustre comment un expéditeur peut compresser un fichier HTML à l'aide d'un encodage de contenu et envoyer les données fragmentées à l'aide d'un encodage de transfert. Le processus

« reconstruire » le corps est inversé sur le récepteur.

Règles de codage de transfert

Lorsqu'un codage de transfert est appliqué au corps d'un message, quelques règles doivent être respectées :

• L'ensemble des codages de transfert doit inclure le « chunked ». La seule exception est si le message est terminé par la fermeture de la connexion.

• Lorsque l'encodage de transfert fragmenté est utilisé, il doit s'agir du dernier encodage de transfert appliqué au corps du message.

• L'encodage de transfert fragmenté ne doit pas être appliqué à un corps de message plus d'une fois.

* L'en-tête Trailer a été ajouté après que l'encodage fragmenté initial a été ajouté aux brouillons de la spécification HTTP/1.1, de sorte que certaines applications peuvent ne pas le comprendre (ou ne pas comprendre les bandes-annonces) même si elles prétendent être conformes à HTTP/1.1.

Figure 15-7. Combinaison du codage de contenu et du codage de transfert

Ces règles permettent au destinataire de déterminer la longueur de transfert du message.

Les codages de transfert sont une fonctionnalité relativement récente de HTTP, introduite dans la version 1.1. Les serveurs qui implémentent ces codages doivent veiller particulièrement à ne pas envoyer de messages codés à des applications non HTTP/1.1. De même, si un serveur reçoit un message codé qu'il ne comprend pas, il doit répondre avec le code d'état 501 Non implémenté. Cependant, toutes les applications HTTP/1.1 doivent au moins prendre en charge le codage fragmenté.

Instances variables dans le temps

Les objets Web ne sont pas statiques. Une même URL peut, au fil du temps, pointer vers différentes versions d'un objet. Prenons l'exemple de la page d'accueil de CNN : consulter « http://www.cnn.com » plusieurs fois par jour risque d'afficher une page légèrement différente à chaque fois.

Considérez la page d'accueil de CNN comme un objet et ses différentes versions comme des instances distinctes de cet objet (voir Figure 15-8). Le client de la figure demande la même ressource (URL) plusieurs fois, mais il obtient différentes instances de la ressource au fil du temps. Aux instants (a) et (b), il a la même instance ; à l'instant (c), il a une instance différente.

Le protocole HTTP spécifie des opérations pour une classe de requêtes et de réponses, appelées manipulations d'instances, qui agissent sur les instances d'un objet. Les deux principales méthodes de manipulation d'instances sont les requêtes de plage et l'encodage delta. Ces deux méthodes nécessitent que les clients soient capables d'identifier la copie exacte de la ressource dont ils disposent (le cas échéant) et de demander de nouvelles instances de manière conditionnelle. Ces mécanismes sont abordés plus loin dans ce chapitre.

Instances variables dans le temps

Figure 15-8. Les instances sont des instantanés d'une ressource dans le temps.

Validateurs et fraîcheur

Revenez à la figure 15-8. Le client ne dispose pas initialement d'une copie de la ressource ; il envoie donc une requête au serveur pour la lui demander. Le serveur répond avec la version 1 de la ressource. Le client peut alors mettre cette copie en cache, mais pour combien de temps ?

Une fois le document expiré chez le client (c'est-à-dire lorsqu'il ne peut plus considérer sa copie comme valide), il doit en demander une

nouvelle copie au serveur. En revanche, si le document n'a pas été modifié sur le serveur, le client n'a pas besoin de le recevoir à nouveau ; il peut simplement continuer à utiliser sa copie en cache.

Cette requête spéciale, appelée requête conditionnelle, exige que le client indique au serveur sa version actuelle, à l'aide d'un validateur, et demande l'envoi d'une copie uniquement si sa version actuelle n'est plus valide. Examinons plus en détail les trois concepts clés : fraîcheur, validateurs et conditions.

Fraîcheur

Les serveurs doivent fournir aux clients des informations sur la durée pendant laquelle ils peuvent mettre en cache leur contenu et le considérer comme récent. Les serveurs peuvent fournir ces informations via l'un des deux en-têtes suivants : Expires et Cache-Control.

L'en-tête Expires spécifie la date et l'heure exactes d'expiration du document, c'est-à-dire lorsqu'il ne peut plus être considéré comme récent. La syntaxe de l'en-tête Expires est la suivante :

Expire : dim. 18 mars 2001 23:59:59 GMT

Pour qu'un client et un serveur utilisent correctement l'en-tête Expires, leurs horloges doivent être synchronisées. Ce n'est pas toujours simple, car aucun des deux n'exécute un protocole de synchronisation d'horloge tel que le protocole NTP (Network Time Protocol). Un mécanisme définissant l'expiration en temps relatif est plus utile. L'en-tête Cache-Control permet de spécifier l'âge maximal d'un document en secondes, c'est-à-dire le temps total écoulé depuis que le document a quitté le serveur. L'âge ne dépend pas de la synchronisation d'horloge et est donc susceptible de produire des résultats plus précis.

L'en-tête Cache-Control est très puissant. Il peut être utilisé par les serveurs et les clients pour décrire la fraîcheur des données, en utilisant des directives plus complètes que la simple spécification d'un âge ou d'une date d'expiration. Le tableau 15-3 répertorie certaines des directives pouvant accompagner l'en-tête Cache-Control.

Tableau 15-3. Directives d'en-tête Cache-Control

Directive Type de message Description

Demande sans cache Ne renvoyez pas une copie en cache du document sans l'avoir d'abord revalidé auprès du serveur.

Demande sans stockage Ne pas renvoyer de copie en cache du document. Ne pas stocker la réponse du serveur.

Demande max-age Le document dans le cache ne doit pas être plus ancien que l'âge spécifié.

Demande max-stale Le document peut être obsolète en fonction des informations d'expiration spécifiées par le serveur, mais il ne doit pas avoir expiré depuis plus longtemps que la valeur de cette directive.

Requête min-fresh L'âge du document ne doit pas être supérieur à son âge plus la quantité spécifiée. En d'autres termes, la réponse doit être fraîche pendant au moins la durée spécifiée.

Demande de non-transformation Le document ne doit pas être transformé avant d'être envoyé.

Demande only-if-cached Envoyer le document uniquement s'il est dans le cache, sans contacter le serveur d'origine.

Réponse publique La réponse peut être mise en cache par n'importe quel cache.

Réponse privée La réponse peut être mise en cache de manière à ce qu'elle ne soit accessible que par un seul client.

Réponse no-cache : si la directive est accompagnée d'une liste de champs d'en-tête, le contenu peut être mis en cache et servi aux clients, mais les champs d'en-tête répertoriés doivent d'abord être supprimés. Si aucun champ d'en-tête n'est spécifié, la copie mise en cache ne doit pas être servie sans revalidation auprès du serveur.

no-store Réponse La réponse ne doit pas être mise en cache.

Réponse sans transformation La réponse ne doit pas être modifiée de quelque manière que ce soit avant d'être servie.

must-revalidate Réponse La réponse doit être revalidée auprès du serveur avant d'être servie.

proxy-revalidate Response : les caches partagés doivent revalider la réponse auprès du serveur d'origine avant de la servir. Cette directive peut être ignorée par les caches privés.

Réponse max-age Spécifie la durée maximale pendant laquelle le document peut être mis en cache et toujours considéré comme récent.

Réponse s-max-age : Spécifie l'âge maximal du document tel qu'il s'applique aux caches partagés (remplaçant la directive max-age, le cas échéant). Cette directive peut être ignorée par les caches privés.

La mise en cache et la fraîcheur ont été abordées plus en détail au chapitre 7.

Conditionnels et validateurs

Lorsqu'une copie du cache est demandée et qu'elle n'est plus récente, le cache doit s'assurer qu'il dispose d'une copie récente. Le cache peut récupérer la copie actuelle depuis le serveur d'origine, mais dans de nombreux cas, le document sur le serveur est toujours identique à la

copie obsolète du cache. Nous l'avons vu dans la figure 15-8b : la copie en cache peut avoir expiré, mais le

Validateurs et fraîcheur

Le contenu du serveur reste identique à celui du cache. Si un cache récupère systématiquement le document d'un serveur, même s'il est identique à la copie expirée du cache, il gaspille la bande passante réseau, impose une charge inutile au cache et au serveur, et ralentit l'ensemble du processus.

Pour résoudre ce problème, HTTP permet aux clients de demander une copie uniquement si la ressource a été modifiée, grâce à des requêtes spéciales appelées requêtes conditionnelles. Ces requêtes conditionnelles sont des messages HTTP classiques, mais elles ne sont exécutées que si une condition particulière est remplie. Par exemple, un cache peut envoyer le message GET conditionnel suivant à un serveur, lui demandant d'envoyer le fichier /announce.html uniquement si celui-ci a été modifié depuis le 29 juin 2002 (date de la dernière modification du document mis en cache par son auteur) :

OBTENIR /announce.html HTTP/1.0

Si modifié depuis : sam. 29 juin 2002, 14:30:00 GMT

Les requêtes conditionnelles sont implémentées par des en-têtes conditionnels commençant par « If- ». Dans l'exemple ci-dessus, l'en-tête conditionnel est « If-Modified-Since ». Un en-tête conditionnel autorise l'exécution d'une méthode uniquement si la condition est vraie. Si la condition n'est pas vraie, le serveur renvoie un code d'erreur HTTP.

Chaque condition fonctionne sur un validateur particulier. Un validateur est un attribut particulier de l'instance de document testée. Conceptuellement, on peut comparer le validateur au numéro de série, au numéro de version ou à la date de dernière modification d'un document. Un client avisé (figure 15-8b) enverrait une requête de validation conditionnelle au serveur indiquant : « Envoyez-moi la ressource uniquement si elle n'est plus en version 1 ; j'ai la version 1. » Nous avons abordé la revalidation conditionnelle du cache au chapitre 7, mais nous étudierons plus en détail les validateurs d'entités dans ce chapitre.

L'en-tête conditionnel If-Modified-Since teste la date de dernière modification d'une instance de document ; on dit donc que cette date est le validateur. L'en-tête conditionnel If-None-Match teste la valeur ETag d'un document, qui est un mot-clé spécial ou une balise d'identification de version associée à l'entité. Last-Modified et ETag sont les deux principaux validateurs utilisés par HTTP. Le tableau 15-4 répertorie quatre en-têtes HTTP utilisés pour les requêtes

conditionnelles. Le type de validateur utilisé est indiqué à côté de chaque en-tête conditionnel.

Tableau 15-4. Types de requêtes conditionnelles

Type de demande Validateur Description

If-Modified-Since Last-Modified Envoyez une copie de la ressource si la version qui a été modifiée pour la dernière fois à ce moment-là dans votre en-tête de réponse Last-Modified précédent n'est plus la plus récente.

Si-non modifié-depuis la dernière modification Envoyez une copie de la ressource uniquement si elle est identique à la version qui a été modifiée pour la dernière fois à ce moment-là dans votre en-tête de réponse Last-Modified précédent.

If-Match ETag Envoyez une copie de la ressource si sa balise d'entité est la même que celle de votre en-tête de réponse ETag précédent.

If-None-Match ETag Envoyez une copie de la ressource si sa balise d'entité est différente de celle de votre en-tête de réponse ETag précédent.

HTTP classe les validateurs en deux classes : les validateurs faibles et les validateurs forts. Les validateurs faibles n'identifient pas toujours de manière unique une instance d'une ressource ; les validateurs forts doivent le faire. La taille de l'objet en octets est un exemple de validateur faible. Le contenu de la ressource peut changer même si sa taille reste la même ; un validateur hypothétique basé sur le nombre d'octets n'indique donc qu'une faible modification. Une somme de contrôle cryptographique du contenu de la ressource (comme MD5) est cependant un validateur fort ; elle change lorsque le document est modifié.

L'heure de dernière modification est considérée comme un validateur faible car, bien qu'elle indique l'heure de dernière modification de la ressource, elle l'indique avec une précision d'au plus une seconde. Étant donné qu'une ressource peut changer plusieurs fois par seconde et que les serveurs peuvent traiter des milliers de requêtes par seconde, la date de dernière modification peut ne pas toujours refléter les changements. L'en-tête ETag est considéré comme un validateur fort, car le serveur peut y placer une valeur distincte à chaque modification. Les numéros de version et les sommes de contrôle condensées sont de bons candidats pour l'en-tête ETag, mais ils peuvent contenir n'importe quel texte. Les en-têtes ETag sont flexibles ; ils acceptent des valeurs textuelles arbitraires (« balises ») et permettent de concevoir diverses stratégies de validation client et serveur.

Les clients et les serveurs peuvent parfois souhaiter adopter une version plus souple de la validation des balises d'entité. Par exemple,

un serveur peut vouloir apporter des modifications esthétiques à un document volumineux et populaire en cache sans déclencher de transfert de masse lors de la revalidation des caches. Dans ce cas, le serveur peut annoncer une balise d'entité « faible » en la préfixant par « W/ ». Une balise d'entité faible ne doit être modifiée que lorsque l'entité associée change de manière significative sur le plan sémantique. Une balise d'entité forte doit être modifiée dès que la valeur de l'entité associée change, quelle qu'en soit la nature.

L'exemple suivant montre comment un client peut revalider auprès d'un serveur à l'aide d'une balise d'entité faible. Le serveur ne renvoie un corps que si le contenu du document a changé de manière significative depuis la version 4.0 :

OBTENIR /announce.html HTTP/1.1

Si-Aucune-Correspondance : W/« v4.0 »

En résumé, lorsque les clients accèdent plusieurs fois à la même ressource, ils doivent d'abord déterminer si leur copie actuelle est toujours à jour. Dans le cas contraire, ils doivent obtenir la dernière version auprès du serveur. Pour éviter de recevoir une copie identique si la ressource n'a pas changé, les clients peuvent envoyer des requêtes conditionnelles au serveur, en spécifiant des validateurs identifiant de manière unique leurs copies actuelles. Les serveurs n'enverront alors une copie de la ressource que si elle est différente de celle du client. Pour plus de détails sur la revalidation du cache, reportez-vous à la section « Étapes de traitement du cache » du chapitre 7.

Demandes de portée

Nous comprenons maintenant comment un client peut demander à un serveur de lui envoyer une ressource uniquement si sa copie n'est plus valide. HTTP va plus loin : il permet aux clients de demander uniquement une partie ou une partie d'un document.

Demandes de portée

Imaginez que vous soyez aux trois quarts du téléchargement du dernier logiciel à la mode via une connexion modem lente et qu'une panne réseau interrompe votre connexion. Vous auriez attendu un certain temps avant de devoir tout recommencer, en espérant que la même chose ne se reproduise plus.

Avec les requêtes de plage, un client HTTP peut reprendre le téléchargement d'une entité en demandant la plage ou la partie de l'entité qu'il n'a pas pu obtenir (à condition que l'objet n'ait pas changé sur le serveur d'origine entre la première requête et la requête de plage suivante). Par exemple :

OBTENIR /bigfile.html HTTP/1.1

Hébergeur : www.joes-hardware.com

Plage : octets = 4 000-

Agent utilisateur : Mozilla/4.61 [fr] (WinNT; I) ...

Dans cet exemple, le client demande le reste du document après les 4 000 premiers octets (il n'est pas nécessaire de préciser les octets de fin, car la taille du document peut être inconnue du demandeur). Les requêtes de plage de ce type peuvent être utilisées en cas d'échec de la requête, lorsque le client a reçu les 4 000 premiers octets avant l'échec. L'en-tête Plage permet également de demander plusieurs plages (les plages peuvent être spécifiées dans n'importe quel ordre et peuvent se chevaucher). Par exemple, imaginez un client se connectant simultanément à plusieurs serveurs et demandant différentes plages du même document à partir de différents serveurs afin d'accélérer le téléchargement global du document. Si les clients demandent plusieurs plages dans une seule requête, les réponses reviennent sous la forme d'une seule entité, avec un corps en plusieurs parties et un en-tête Content-Type: multipart/byteranges.

Tous les serveurs n'acceptent pas les requêtes de plage, mais beaucoup le font. Les serveurs peuvent signaler aux clients qu'ils acceptent les plages en incluant l'en-tête « Accept-Ranges » dans leurs réponses.

La valeur de cet en-tête est l'unité de mesure, généralement des octets.* Par exemple :

HTTP/1.1 200 OK

Date : vendredi 5 novembre 1999 22:35:15 GMT

Serveur : Apache/1.2.4

Accept-Ranges : octets ...

La figure 15-9 montre un exemple d’un ensemble de transactions HTTP impliquant des plages.

Les en-têtes de plage sont largement utilisés par les logiciels clients de partage de fichiers peer-to-peer populaires pour télécharger simultanément différentes parties de fichiers multimédias, à partir de différents homologues.

Notez que les requêtes de plage sont une classe de manipulations d'instances, car elles constituent des échanges entre un client et un serveur pour une instance particulière d'un objet. Autrement dit, une requête de plage d'un client n'a de sens que si le client et le serveur ont la même version d'un document.

* La spécification HTTP/1.1 définit uniquement le jeton d'octets, mais les implémenteurs de serveur et de client pourraient proposer leurs propres unités pour mesurer ou découper une entité.

Figure 15-9. Exemple de requête de plage d'entités

Codage Delta

Nous avons décrit les différentes versions d'une page web comme des instances distinctes. Si un client possède une copie expirée d'une page, il demande la dernière instance. Si le serveur possède une instance plus récente, il la transmet au client, et ce dernier lui envoie la nouvelle instance complète, même si seule une petite partie de la page a été modifiée.

Plutôt que de lui envoyer la nouvelle page entière, le client obtiendrait la page plus rapidement si le serveur envoyait uniquement les modifications à sa copie (à condition que le nombre de modifications soit faible). L'encodage delta est une extension du protocole HTTP qui optimise les transferts en communiquant les modifications plutôt que des objets entiers. L'encodage delta est un type de manipulation d'instance, car il repose sur l'échange d'informations entre clients et serveurs sur des instances particulières d'un objet. La RFC 3229 décrit l'encodage delta.

La figure 15-10 illustre plus clairement le mécanisme de demande, de génération, de réception et d'application d'un document codé par delta. Le client doit indiquer au serveur la version de la page dont il dispose, sa volonté d'accepter un delta de la dernière version de la page et les algorithmes qu'il connaît pour appliquer ces deltas à sa version actuelle.

Codage Delta

Le serveur doit vérifier s'il dispose de la version de la page du client et comment calculer les deltas entre la dernière version et la version du client (plusieurs algorithmes permettent de calculer la différence entre deux objets). Il doit ensuite calculer le delta, l'envoyer au client, l'informer de l'envoi d'un delta et spécifier le nouvel identifiant de la dernière version de la page (car c'est la version que le client obtiendra après avoir appliqué le delta à son ancienne version).

Figure 15-10. Mécanique du codage delta

Le client utilise l'identifiant unique de sa version de la page (envoyé par le serveur dans sa réponse précédente, dans l'en-tête ETag), dans un en-tête If-None-Match. Il indique ainsi au serveur si la dernière version de la page est disponible.

n'a pas le même ETag, envoyez-moi la dernière version de la page. » Seul l'en-tête IfNone-Match obligerait alors le serveur à envoyer au client la dernière version complète de la page (si elle était différente de la version du client).

Le client peut toutefois indiquer au serveur qu'il accepte un delta de la page en lui envoyant également un en-tête A-IM. A-IM est l'abréviation de Accept-Instance-Manipulation (« Au fait, j'accepte certaines formes de manipulation d'instance, donc si vous en appliquez une, vous n'aurez pas à m'envoyer le document complet. »). Dans l'en-tête A-IM, le client spécifie les algorithmes qu'il sait appliquer pour générer la dernière version d'une page à partir d'une ancienne version et d'un delta. Le serveur renvoie les éléments suivants : un code de réponse spécial (226 IM Used) indiquant au client qu'il lui envoie une manipulation d'instance de l'objet demandé, et non l'objet complet ; un en-tête IM (abréviation de Instance-Manipulation), qui spécifie l'algorithme utilisé pour calculer le delta ; le nouvel en-tête ETag ; et un en-tête Delta-Base, qui spécifie l'ETag du document utilisé comme base pour le calcul du delta (idéalement, le même que l'ETag de la requête If-None-Match du client !). Les en-têtes utilisés dans le codage delta sont résumés dans le tableau 15-5.

Tableau 15-5. En-têtes de codage Delta

Description de l'en-tête

ETag : identifiant unique pour chaque instance d'un document. Envoyé par le serveur dans la réponse ; utilisé par les clients dans les requêtes ultérieures dans les en-têtes If-Match et If-None-Match.

En-tête de requête If-None-Match envoyé par le client, demandant au serveur un document si et seulement si la version du document du client est différente de celle du serveur.

En-tête de demande du client A-IM indiquant les types de manipulations d'instance acceptées.

En-tête de réponse du serveur de messagerie instantanée spécifiant le type de manipulation d'instance appliqué à la réponse. Cet en-tête est envoyé lorsque le code de réponse est 226 IM utilisé.

En-tête de réponse du serveur Delta-Base qui spécifie l'ETag du document de base utilisé pour générer le delta (doit être le même que l'ETag dans l'en-tête If-None-Match de la demande client).

Manipulations d'instances, générateurs Delta et applicateurs Delta

Les clients peuvent spécifier les types de manipulation d'instance qu'ils acceptent grâce à l'en-tête A-IM. Les serveurs précisent le type de manipulation d'instance utilisé dans l'en-tête IM. Quels sont les types de manipulation d'instance acceptés et à quoi servent-ils ? Le tableau 15-6 répertorie certains types de manipulation d'instance enregistrés par l'IANA.

Tableau 15-6. Types de manipulations d'instances enregistrés par l'IANA

Description du type

vcdiff Delta utilisant l'algorithme vcdiff

diff Delta en utilisant la commande Unix diff -e

gdiff Delta utilisant l'algorithme gdiffb

Codage Delta

Tableau 15-6. Types de manipulations d'instances enregistrés par l'IANA (suite)

Description du type

Compression gzip à l'aide de l'algorithme gzip

Compression de dégonflage à l'aide de l'algorithme de dégonflage

plage Utilisé dans une réponse de serveur pour indiquer que la réponse est un contenu partiel résultant d'une sélection de plage

identité Utilisé dans l'en-tête A-IM d'une demande client pour indiquer que le client est prêt à accepter une manipulation d'instance d'identité

Un projet Internet (brouillon-korn-vcdiff-01) décrit l'algorithme vcdiff. Cette spécification a été approuvée par l'IESG début 2002 et devrait être publiée sous forme de RFC prochainement. b http://www.w3org/TR/NOTE-gdiff-19970901.html décrit l'algorithme GDIFF.

Un générateur de delta sur le serveur, comme illustré à la figure 15-10, prend le document de base et sa dernière instance et calcule le delta entre les deux à l'aide de l'algorithme spécifié par le client dans l'en-tête A-IM. Côté client, un applicateur de delta prend le delta et l'applique au document de base pour générer sa dernière instance. Par exemple, si l'algorithme utilisé pour générer le delta est la commande Unix diff -e, le client peut appliquer le delta à l'aide de l'éditeur de texte Unix ed, car diff -e <file1> <file2> génère l'ensemble des commandes ed qui convertiront <file1> en <file2>. ed est un éditeur très simple prenant en charge quelques commandes. Dans l'exemple de la figure 15-10, 5c indique de supprimer la ligne 5 du document de base, et chisels.<cr>. indique d'ajouter « chisels. ». C'est tout. Des instructions plus complexes peuvent être générées pour des modifications plus importantes. L'algorithme Unix diff -e compare les fichiers ligne par ligne. Cela convient évidemment aux fichiers texte, mais ne fonctionne pas pour les fichiers binaires. L'algorithme vcdiff est plus puissant, fonctionnant même pour les fichiers non texte et produisant généralement des écarts plus faibles que diff -e.

La spécification d'encodage delta définit en détail le format des en-têtes A-IM et IM. Il est important de noter que plusieurs manipulations d'instances peuvent être spécifiées dans ces en-têtes (avec les valeurs de qualité correspondantes). Les documents peuvent subir plusieurs manipulations d'instances avant d'être renvoyés aux clients, afin

d'optimiser la compression. Par exemple, les deltas générés par l'algorithme vcdiff peuvent à leur tour être compressés à l'aide de l'algorithme gzip. La réponse du serveur contiendrait alors l'en-tête IM: vcdiff, gzip. Le client compresserait d'abord le contenu, puis appliquerait les résultats du delta à sa page de base afin de générer le document final.

L'encodage delta peut réduire les temps de transfert, mais sa mise en œuvre peut s'avérer complexe. Imaginez une page fréquemment modifiée et consultée par de nombreuses personnes. Un serveur prenant en charge l'encodage delta doit conserver toutes les copies de cette page au fil de ses modifications, afin de déterminer les modifications entre la copie d'un client demandeur et la dernière. (Si le document change fréquemment, les clients qui le demandent en obtiendront des instances différentes. Lors de leurs requêtes ultérieures au serveur, ils demanderont des modifications entre leur instance et la dernière. Pour ne leur transmettre que les modifications, le serveur doit conserver des copies de toutes les instances précédentes des clients.) En échange d'une latence réduite lors de la diffusion des documents, les serveurs doivent augmenter l'espace disque nécessaire pour conserver les anciennes instances. L'espace disque supplémentaire nécessaire à cette fin peut rapidement annuler les avantages des volumes de transfert réduits.

Pour plus d'informations

Pour plus d'informations sur les entités et les encodages, voir :

http://www.ietf.org/rfc/rfc2616.txt

La spécification HTTP/1.1, RFC 2616, est la principale référence pour la gestion et les codages des corps d'entité.

http://www.ietf.org/rfc/rfc3229.txt

La RFC 3229, « Encodage delta dans HTTP », décrit comment l'encodage delta peut être pris en charge en tant qu'extension de HTTP/1.1.

Introduction à la compression de données

Khalid Sayood, Morgan Kaufmann Publishers. Ce livre explique certains algorithmes de compression pris en charge par les encodages de contenu HTTP.

http://www.ietf.org/rfc/rfc1521.txt

La RFC 1521, « Extensions de messagerie Internet polyvalentes, première partie : Mécanismes de spécification et de description du format des corps de messages Internet », décrit le format des corps MIME. Ce document de référence est utile car HTTP s'inspire largement de MIME. Ce document est notamment conçu pour fournir des fonctionnalités permettant d'inclure plusieurs objets dans un même

message, de représenter le corps du texte dans des jeux de caractères autres que l'US-ASCII, de représenter des messages texte multi-polices formatés et de représenter des éléments non textuels tels que des images et des fragments audio.

http://www.ietf.org/rfc/rfc2045.txt

La RFC 2045, « Extensions de messagerie Internet polyvalentes, première partie : format des corps de messages Internet », spécifie les différents en-têtes utilisés pour décrire la structure des messages MIME, dont beaucoup sont similaires ou identiques à HTTP.

http://www.ietf.org/rfc/rfc1864.txt

La RFC 1864, « Le champ d'en-tête Content-MD5 », fournit des détails historiques sur le comportement et l'utilisation prévue du champ d'en-tête Content-MD5 dans le contenu MIME comme contrôle d'intégrité du message.

http://www.ietf.org/rfc/rfc3230.txt

La RFC 3230, « Instance Digests in HTTP », décrit les améliorations apportées à la gestion des entitydigest HTTP qui corrigent les faiblesses présentes dans la formulation Content-MD5.

Pour plus d'informations

Chapitre 16Ceci est le titre du livreCHAPITRE 16

Internationalisation

Chaque jour, des milliards de personnes rédigent des documents dans des centaines de langues. Pour concrétiser la vision d'un Web véritablement mondial, HTTP doit prendre en charge le transport et le traitement de documents internationaux, dans de nombreuses langues et alphabets.

Ce chapitre aborde deux problématiques majeures de l'internationalisation du Web : les codages de jeux de caractères et les balises de langue. Les applications HTTP utilisent des codages de jeux de caractères pour demander et afficher du texte dans différents alphabets, et utilisent des balises de langue pour décrire et restreindre le contenu aux langues comprises par l'utilisateur. Nous terminons par une brève discussion sur les URI multilingues et les dates.

Ce chapitre :

• Explique comment HTTP interagit avec les schémas et les normes pour les alphabets multilingues

• Donne un aperçu rapide de la terminologie, de la technologie et des normes pour aider les programmeurs HTTP à faire les choses correctement (les lecteurs familiarisés avec les codages de caractères peuvent ignorer cette section)

• Explique le système de nommage standard des langues et comment les balises de langue standardisées décrivent et sélectionnent le contenu

• Décrit les règles et les précautions pour les URI internationaux

• Discute brièvement des règles relatives aux dates et d'autres questions d'internationalisation

Prise en charge HTTP pour le contenu international

Les messages HTTP peuvent véhiculer du contenu dans n'importe quelle langue, tout comme ils peuvent véhiculer des images, des films ou tout autre type de média. Pour HTTP, le corps de l'entité n'est qu'un ensemble de bits.

Pour prendre en charge le contenu international, les serveurs doivent informer les clients de l'alphabet et des langues de chaque document, afin que le client puisse correctement décompresser les éléments du document en caractères et traiter et présenter correctement le contenu à l'utilisateur.

Les serveurs indiquent aux clients l'alphabet et la langue d'un document grâce au paramètre de jeu de caractères HTTP Content-Type et aux en-têtes Content-Language. Ces en-têtes décrivent le contenu du corps de l'entité, comment convertir le contenu en caractères appropriés pour l'affichage à l'écran et la langue parlée représentée par les mots.

Parallèlement, le client doit indiquer au serveur les langues comprises par l'utilisateur et les algorithmes de codage alphabétique installés par le navigateur. Le client envoie les en-têtes Accept-Charset et Accept-Language pour indiquer au serveur les algorithmes de codage de jeux de caractères et les langues compris par le client, ainsi que leurs préférences.

Les en-têtes HTTP Accept suivants peuvent être envoyés par un francophone qui préfère sa langue maternelle (mais parle un peu anglais à la rigueur) et qui utilise un navigateur prenant en charge l'encodage de jeu de caractères d'Europe de l'Ouest iso-8859-1 et l'encodage de jeu de caractères Unicode UTF-8 :

Accepter-Langue : fr, en ; q = 0,8

Accept-Charset : iso-8859-1, utf-8

Le paramètre « q=0,8 » est un facteur de qualité, donnant une priorité moindre à l'anglais (0,8) qu'au français (1,0 par défaut).

Jeux de caractères et HTTP

Alors, passons directement aux aspects les plus importants (et les plus déroutants) de l'internationalisation du Web : les scripts alphabétiques internationaux et leurs codages de jeux de caractères.

Les normes de jeux de caractères web peuvent être assez déroutantes. Nombreux sont ceux qui se laissent aller à la frustration lorsqu'ils tentent de développer un logiciel web international, en raison d'une terminologie complexe et incohérente, de documents de normes payants et d'une méconnaissance des langues étrangères. Cette section et la suivante devraient vous faciliter l'utilisation des jeux de caractères avec HTTP.

Charset est un codage de caractères en bits

Les valeurs de jeu de caractères HTTP indiquent comment convertir les bits du contenu d'une entité en caractères d'un alphabet spécifique. Chaque balise de jeu de caractères désigne un algorithme permettant de traduire les bits en caractères (et inversement). Les balises de jeu de caractères sont normalisées dans le registre des jeux de caractères MIME, géré par l'IANA (voir http://www.iana.org/assignments/character-sets). L'annexe H en résume un grand nombre.

L'en-tête Content-Type suivant indique au récepteur que le contenu est un fichier HTML, et le paramètre charset indique au récepteur d'utiliser le schéma de décodage du jeu de caractères arabes iso-8859-6 pour décoder les bits de contenu en caractères :

Type de contenu : texte/html ; jeu de caractères = iso-8859-6

Le schéma de codage iso-8859-6 mappe les valeurs de 8 bits dans les alphabets latin et arabe, y compris les chiffres, la ponctuation et d'autres symboles.* Par exemple, dans la Figure 16-1, le modèle de bits en surbrillance a la valeur de code 225, qui (selon iso-8859-6) correspond à la lettre arabe « FEH » (un son semblable à la lettre anglaise « F »).

Figure 16-1. Le paramètre charset indique au client comment passer des bits aux caractères.

Certains codages de caractères (par exemple, UTF-8 et iso-2022-jp) sont des codes plus complexes, à longueur variable, dont le nombre de bits par caractère varie. Ce type de codage permet d'utiliser des bits supplémentaires pour prendre en charge les alphabets comportant un grand nombre de caractères (comme le chinois et le japonais), tout en utilisant moins de bits pour prendre en charge les caractères latins standard.

Comment fonctionnent les jeux de caractères et les codages

Voyons ce que font réellement les jeux de caractères et les encodages.

Nous souhaitons convertir les bits d'un document en caractères affichables à l'écran. Cependant, comme il existe de nombreux alphabets et méthodes de codage des caractères en bits (chacune présentant des avantages et des inconvénients), nous avons besoin d'une méthode standard pour décrire et appliquer l'algorithme de décodage bits-caractères.

Les conversions de bits en caractères se déroulent en deux étapes, comme illustré dans la figure 16-2 :

• Dans la figure 16-2a, les bits d'un document sont convertis en un code de caractère identifiant un caractère numéroté particulier dans un jeu de caractères codés particulier. Dans l'exemple, le code de caractère décodé est le numéro 225.

• Dans la figure 16-2b, le code de caractère permet de sélectionner un élément particulier du jeu de caractères codés. Dans la norme ISO-8859-6, la valeur 225 correspond à la « LETTRE ARABE FEH ». Les algorithmes utilisés aux étapes a et b sont déterminés à partir des

Balise de jeu de caractères MIME.

L'un des objectifs clés des systèmes de caractères internationalisés est d'isoler la sémantique (lettres) de la présentation (formes de présentation graphique). HTTP s'intéresse à

* Contrairement au chinois et au japonais, l'arabe ne comporte que 28 caractères. Huit bits fournissent 256 valeurs uniques, ce qui laisse suffisamment de place aux caractères latins, arabes et autres symboles utiles.

Figure 16-2. HTTP « charset » combine un schéma de codage de caractères et un jeu de caractères codés

Il s'agit uniquement du transport des données de caractères et des étiquettes de langue et de jeu de caractères associées. La présentation des formes de caractères est gérée par le logiciel d'affichage graphique de l'utilisateur (navigateur, système d'exploitation, polices), comme illustré à la figure 16-2c.

Le mauvais jeu de caractères donne les mauvais caractères

Si le client utilise un paramètre de jeu de caractères incorrect, il affichera des caractères étranges et erronés. Imaginons qu'un navigateur obtienne la valeur 225 (binaire 11100001) dans le corps :

• Si le navigateur pense que le corps du message est codé avec les codes de caractères d'Europe occidentale ISO-8859-1, il affichera un « a » latin minuscule avec un accent aigu :

• Si le navigateur utilise les codes arabes ISO-8859-6, il affichera « FEH » :

• Si le navigateur utilise le grec iso-8859-7, il affichera un petit « Alpha » :

• Si le navigateur utilise les codes hébreux iso-8859-8, il affichera « BET » :

Valeurs de jeu de caractères MIME standardisées

La combinaison d'un codage de caractères particulier et d'un jeu de caractères codés particulier est appelée jeu de caractères MIME. HTTP utilise des balises de jeu de caractères MIME standardisées dans les en-têtes Content-Type et Accept-Charset. Les valeurs de jeu de caractères MIME sont enregistrées auprès de l'IANA.* Le tableau 16-1 répertorie quelques schémas de codage de jeu de caractères MIME utilisés par les documents et les navigateurs. Une liste plus complète est fournie à l'annexe H.

Tableau 16-1. Balises d'encodage du jeu de caractères MIME

Valeur du jeu de caractères MIME Description

US-ASCII : le célèbre codage de caractères normalisé en 1968 sous le nom ANSI_X3.4-1968. Également appelé ASCII, le préfixe « US » est privilégié en raison de plusieurs variantes internationales de la norme ISO 646 qui modifient certains caractères. US-ASCII convertit les valeurs de 7 bits en 128 caractères. Le bit de poids fort est inutilisé.

iso-8859-1 est une extension 8 bits de l'ASCII prenant en charge les langues d'Europe occidentale. Elle utilise le bit de poids fort pour inclure de nombreux caractères d'Europe occidentale, tout en conservant les codes ASCII (0-127). Également appelée iso-latin-1, ou surnommée « Latin1 », elle est également utilisée pour la traduction des caractères.

ISO-8859-2 : étend l'ASCII aux caractères des langues d'Europe centrale et orientale, notamment le tchèque, le polonais et le roumain. Également appelé iso-latin-2.

iso-8859-5 Étend l'ASCII pour inclure les caractères cyrilliques, pour les langues telles que le russe, le serbe et le bulgare.

La norme ISO-8859-6 étend l'ASCII aux caractères arabes. La forme des caractères arabes variant selon leur position dans un mot, l'arabe

nécessite un moteur d'affichage qui analyse le contexte et génère la forme correcte pour chaque caractère.

ISO-8859-7 : étend l'ASCII aux caractères grecs modernes. Anciennement connu sous le nom ELOT-928 ou ECMA-118:1986.

iso-8859-8 Étend l'ASCII pour inclure les caractères hébreux et yiddish.

iso-8859-15 Mises à jour iso-8859-1, remplaçant certains symboles de ponctuation et de fraction moins nécessaires par des symboles oubliés

Lettres françaises et finlandaises, et remplacement du symbole monétaire international par le symbole de la nouvelle monnaie, l'euro. Ce jeu de caractères, surnommé « Latin0 », pourrait un jour remplacer l'ISO-8859-1 comme jeu de caractères par défaut en Europe.

iso-2022-jp est un codage largement utilisé pour les e-mails et le contenu web en japonais. Il s'agit d'un schéma de codage à longueur variable qui prend en charge les caractères ASCII mono-octets, mais utilise des séquences d'échappement modales de trois caractères pour passer à trois jeux de caractères japonais différents.

euc-jp est un codage à longueur variable conforme à la norme ISO 2022 qui utilise des séquences binaires explicites pour identifier chaque caractère, sans nécessiter de modes ni de séquences d'échappement. Il utilise des séquences de caractères de 1, 2 et 3 octets pour identifier les caractères de plusieurs jeux de caractères japonais.

Shift_JIS : cet encodage a été initialement développé par Microsoft et est parfois appelé SJIS ou MS Kanji. Il est un peu complexe, pour des raisons de compatibilité historique, et ne peut pas mapper tous les caractères, mais il reste courant.

* Voir http://www.iana.org/numbers.htm pour la liste des valeurs de jeu de caractères enregistrées.

Tableau 16-1. Balises d'encodage du jeu de caractères MIME (suite)

Valeur du jeu de caractères MIME Description

koi8-r KOI8-R est un jeu de caractères Internet 8 bits populaire pour le codage russe, défini dans la RFC 1489 de l'IETF. Les initiales sont des translittérations de l'acronyme de « Code for Information Exchange, 8 bit, Russian ».

UTF-8 est un système d'encodage de caractères à longueur variable courant pour représenter l'UCS (Unicode), le jeu de caractères universel. UTF-8 utilise un codage à longueur variable pour les valeurs de code de caractère, chaque caractère étant représenté par un à six octets. L'une des principales caractéristiques d'UTF-8 est sa rétrocompatibilité avec le texte ASCII 7 bits ordinaire.

Microsoft appelle ses jeux de caractères codés « pages de codes ». La page de codes Windows 1252 (également appelée « CP1252 » ou « WinLatin1 ») est une extension de la norme ISO-8859-1.

En-tête de jeu de caractères de type de contenu et balises META

Les serveurs Web envoient au client la balise de jeu de caractères MIME dans l'en-tête Content-Type, en utilisant

le paramètre charset :

Type de contenu : texte/html ; jeu de caractères = iso-2022-jp

Si aucun jeu de caractères n'est explicitement indiqué, le destinataire peut tenter de déduire le jeu de caractères du contenu du document. Pour le contenu HTML, les jeux de caractères peuvent être trouvés dans les balises <META HTTP-EQUIV="Content-Type"> qui décrivent le jeu de caractères.

L'exemple 16-1 montre comment les balises META HTML définissent le jeu de caractères selon le codage japonais iso-2022-jp. Si le document n'est pas au format HTML ou s'il n'y a pas de balise META Content-Type, le logiciel peut tenter de déduire le codage des caractères en analysant le texte réel à la recherche de modèles communs indiquant les langues et les codages.

Exemple 16-1. L'encodage des caractères peut être spécifié dans les balises META HTML.

<TÊTE>

<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-2022-jp">

<META LANG="jp">

<TITLE>Un document japonais</TITLE>

</HEAD>

<CORPS> ...

Si un client ne peut pas déduire un codage de caractères, il suppose iso-8859-1.

L'en-tête Accept-Charset

Il existe des milliers de méthodes d'encodage et de décodage de caractères définies, développées au cours des dernières décennies. La plupart des clients ne prennent pas en charge tous les systèmes d'encodage et de mappage de caractères.

Les clients HTTP peuvent indiquer précisément aux serveurs les systèmes de caractères pris en charge grâce à l'en-tête de requête Accept-Charset. Cette valeur fournit la liste des schémas d'encodage de caractères pris en charge par le client. Par exemple, l'en-tête de

requête HTTP suivant indique qu'un client accepte le système de caractères ISO-8859-1 d'Europe occidentale ainsi que le système de compatibilité Unicode UTF-8 à longueur variable. Un serveur est libre de renvoyer du contenu dans l'un ou l'autre de ces schémas d'encodage.

Accept-Charset : iso-8859-1, utf-8

Notez qu'il n'existe pas d'en-tête de réponse Content-Charset correspondant à l'en-tête de requête Accept-Charset. Le jeu de caractères de la réponse est repris du serveur par le paramètre charset de l'en-tête Content-Type, afin d'être compatible avec

MIME. Dommage que ce ne soit pas symétrique, mais toutes les informations sont toujours là.

Introduction au codage de caractères multilingues

La section précédente décrit comment l'en-tête HTTP Accept-Charset et le paramètre de jeu de caractères Content-Type transmettent les informations de codage des caractères du client et du serveur. Les programmeurs HTTP travaillant fréquemment avec des applications et des contenus internationaux doivent maîtriser les systèmes de caractères multilingues pour comprendre les spécifications techniques et implémenter correctement les logiciels.

Apprendre les systèmes de caractères multilingues n'est pas chose aisée : la terminologie est complexe et incohérente, la lecture des documents de normes est souvent payante et vous n'êtes peut-être pas familier avec les autres langues avec lesquelles vous travaillez. Cette section présente les systèmes de caractères et les normes. Si vous maîtrisez déjà les codages de caractères ou si ce sujet ne vous intéresse pas, n'hésitez pas à passer directement à la section « Balises de langue et HTTP ».

Terminologie des jeux de caractères

Voici huit termes sur les systèmes de caractères électroniques que vous devriez connaître :

Personnage

Lettre alphabétique, chiffre, signe de ponctuation, idéogramme (comme en chinois), symbole ou autre élément textuel de l'écriture. Le jeu de caractères universel (UCS), connu sous le nom informel d'Unicode*, a permis de développer un ensemble normalisé de noms textuels pour de nombreux caractères dans de nombreuses langues, souvent utilisés pour nommer les caractères de manière pratique et unique.†

Glyphe

Motif de trait ou forme graphique unique décrivant un caractère. Un caractère peut avoir plusieurs glyphes s'il peut être écrit de différentes manières (voir figure 16-3).

Caractère codé

Un numéro unique attribué à un personnage afin que nous puissions travailler avec lui.

Espace de codage

Une plage d'entiers que nous prévoyons d'utiliser comme valeurs de code de caractère.

* Unicode est un consortium commercial basé sur UCS qui pilote les produits commerciaux.

† Les noms ressemblent à « LETTRE MAJUSCULE LATINE S » et « LETTRE ARABE QAF ».

Largeur du code

Le nombre de bits dans chaque code de caractère (de taille fixe).

Répertoire de personnages

Un ensemble de caractères particulier (un sous-ensemble de tous les caractères du monde).

jeu de caractères codés

Un ensemble de caractères codés qui utilise un répertoire de caractères (une sélection de caractères du monde entier) et attribue à chaque caractère un code issu d'un espace de codage. Autrement dit, il associe des codes de caractères numériques à des caractères réels.

Schéma de codage des caractères

Algorithme permettant de coder des codes de caractères numériques en une séquence de bits de contenu (et de les décoder). Les schémas de codage de caractères peuvent être utilisés pour réduire la quantité de données nécessaires à l'identification des caractères (compression), contourner les restrictions de transmission et unifier les jeux de caractères codés qui se chevauchent.

Le jeu de caractères est mal nommé

Techniquement, la balise de jeu de caractères MIME (utilisée dans le paramètre de jeu de caractères Content-Type et l'en-tête Accept-Charset) ne spécifie aucun jeu de caractères. La valeur du jeu de caractères MIME désigne un algorithme complet permettant de mapper des bits de données à des codes de caractères uniques. Elle combine les deux concepts distincts de schéma d'encodage de caractères et de jeu de caractères codés (voir Figure 16-2).

Cette terminologie est bâclée et déroutante, car il existe déjà des normes publiées pour les schémas de codage de caractères et pour les jeux de caractères codés.* Voici ce que disent les auteurs de HTTP/1.1 à propos de leur utilisation de la terminologie (dans la RFC 2616) :

Le terme « jeu de caractères » est utilisé dans ce document pour désigner une méthode... permettant de convertir une séquence d'octets en une séquence de caractères... Remarque : cette utilisation du terme « jeu de caractères » est plus communément appelée « encodage de caractères ». Cependant, étant donné que HTTP et MIME partagent le même registre, il est important que la terminologie soit également partagée.

L'IETF adopte également une terminologie non standard dans la RFC 2277 :

Ce document utilise le terme « charset » pour désigner un ensemble de règles de correspondance entre une séquence d'octets et une séquence de caractères, par exemple la combinaison d'un jeu de caractères codés et d'un schéma de codage de caractères ; c'est également ce terme qui est utilisé comme identifiant dans les paramètres MIME « charset= » et enregistré dans le registre des jeux de caractères de l'IANA. (Notez que ce terme n'est PAS utilisé par d'autres organismes de normalisation, comme l'ISO).

Soyez donc vigilant lorsque vous lisez les documents de normes afin de savoir exactement ce qui est défini. Maintenant que nous avons clarifié la terminologie, examinons de plus près les caractères, les glyphes, les jeux de caractères et les codages de caractères.

* Pire encore, la balise de jeu de caractères MIME reprend souvent le nom d'un jeu de caractères codés ou d'un schéma de codage particulier. Par exemple, la norme ISO-8859-1 est un jeu de caractères codés (elle attribue des codes numériques à un ensemble de 256 caractères européens), mais MIME utilise la valeur de jeu de caractères « iso-8859-1 » pour désigner un codage d'identité sur 8 bits du jeu de caractères codés. Cette terminologie imprécise n'est pas fatale, mais lors de la lecture des documents de normes, soyez clair sur les hypothèses.

Personnages

Les caractères sont les éléments de base de l'écriture. Un caractère représente une lettre alphabétique, un chiffre, un signe de ponctuation, un idéogramme (comme en chinois), un symbole mathématique ou toute autre unité d'écriture fondamentale.

Les caractères sont indépendants de la police et du style. La figure 16-3 montre plusieurs variantes du même caractère, appelé « LETTRE MINUSCULE LATINE A ». Un lecteur natif d'une langue d'Europe occidentale reconnaîtrait immédiatement ces cinq formes comme

étant le même caractère, même si les tracés et les styles sont très différents.

Figure 16-3. Un caractère peut avoir plusieurs graphies différentes.

De nombreux systèmes d'écriture utilisent également des formes de trait différentes pour un même caractère, selon sa position dans le mot. Par exemple, les quatre traits de la figure 16-4 représentent tous le caractère « LETTRE ARABE AIN ».* La figure 16-4a illustre l'écriture de « AIN » en tant que caractère autonome. La figure 16-4d montre « AIN » au début d'un mot, la figure 16-4c au milieu d'un mot et la figure 16-4b à la fin d'un mot.†

Figure 16-4. Quatre formes positionnelles du caractère unique « LETTRE ARABE AIN »

Glyphes, ligatures et formes de présentation

Ne confondez pas les caractères avec les glyphes. Les caractères sont les « atomes » uniques et abstraits du langage. Les glyphes sont les façons particulières de dessiner chaque caractère. Chaque caractère

comporte de nombreux glyphes différents, selon le style artistique et l'écriture.‡

De plus, ne confondez pas les caractères avec les formes de présentation. Pour embellir l'écriture, de nombreuses polices et écritures manuscrites permettent de joindre des caractères adjacents en de jolies ligatures, permettant ainsi aux deux caractères de se connecter harmonieusement.

* Le son « AIN » se prononce un peu comme « ayine », mais vers l’arrière de la gorge.

† Notez que les mots arabes s’écrivent de droite à gauche.

‡ Beaucoup utilisent le terme « glyphe » pour désigner l'image bitmap finale, mais techniquement, un glyphe est la forme inhérente à un caractère, indépendamment de la police et du style artistique. Cette distinction n'est ni facile à appliquer, ni utile pour notre propos.

Les typographes joignent souvent « F » et « I » dans une « ligature FI » (voir Figure 16-5a–b), et les écrivains arabes joignent souvent les caractères « LAM » et « ALIF » dans une ligature attrayante (Figure 16-5c–d).

Figure 16-5. Les ligatures sont des formes de présentation stylistique de caractères adjacents, et non de nouveaux caractères.

Voici la règle générale : si le sens du texte change lorsqu'on remplace un glyphe par un autre, les glyphes sont des caractères différents. Sinon, ce sont les mêmes caractères, avec une présentation stylistique différente.*

Jeux de caractères codés

Les jeux de caractères codés, définis dans les RFC 2277 et 2130, associent des entiers à des caractères. Ils sont souvent implémentés sous forme de tableaux†, indexés par numéro de code (voir

Figure 16-6). Les éléments du tableau sont des caractères.‡

Figure 16-6. Les jeux de caractères codés peuvent être considérés comme des tableaux qui associent des codes numériques à des caractères.

Examinons quelques normes importantes de jeux de caractères codés, notamment le jeu de caractères historique US-ASCII, les extensions ISO-8859 à ASCII, le jeu de caractères japonais JIS X 0201 et le jeu de caractères universel (Unicode).

US-ASCII : la mère de tous les jeux de caractères

L'ASCII est le jeu de caractères codés le plus célèbre, normalisé en 1968 sous la norme ANSI X3.4 « American Standard Code for Information Interchange ». L'ASCII utilise

* La distinction entre sémantique et présentation n'est pas toujours claire. Pour faciliter la mise en œuvre, des caractères distincts ont été attribués à certaines variantes de présentation d'un même caractère, mais l'objectif est d'éviter ce problème.

† Les tableaux peuvent être multidimensionnels, de sorte que différents bits du numéro de code indexent différents axes du tableau.

La figure 16-6 utilise une grille pour représenter un jeu de caractères codés. Chaque élément de la grille contient une image de caractère. Ces images sont symboliques. La présence d'une image « D » est une abréviation du caractère « LETTRE MAJUSCULE LATINE D », et non d'un glyphe graphique particulier.

Seules les valeurs de code 0 à 127 sont utilisées ; 7 bits seulement sont donc nécessaires pour couvrir l'espace de code. Le nom privilégié pour l'ASCII est « US-ASCII », afin de le distinguer des variantes internationales du jeu de caractères 7 bits.

Les messages HTTP (en-têtes, URI, etc.) utilisent US-ASCII.

iso-8859

Les normes de jeu de caractères ISO-8859 sont des sur-ensembles 8 bits de l'US-ASCII qui utilisent le bit de poids fort pour ajouter des

caractères pour l'écriture internationale. L'espace supplémentaire offert par ce bit supplémentaire (128 codes supplémentaires) n'est pas suffisant pour contenir tous les caractères européens (sans parler des caractères asiatiques). C'est pourquoi ISO-8859 propose des jeux de caractères personnalisés pour différentes régions :

iso-8859-1 Langues d'Europe occidentale (par exemple, anglais, français)

iso-8859-2 Langues d'Europe centrale et orientale (par exemple, tchèque, polonais)

iso-8859-3 Langues d'Europe du Sud

iso-8859-4 Langues d'Europe du Nord (par exemple, letton, lituanien, groenlandais)

iso-8859-5 cyrillique (par exemple, bulgare, russe, serbe)

iso-8859-6 arabe

iso-8859-7 grec

iso-8859-8 hébreu

iso-8859-9 turc

iso-8859-10 Langues nordiques (par exemple, islandais, inuit)

Modification de la norme ISO-8859-1 qui inclut le nouveau caractère monétaire Euro

L'ISO-8859-1, également appelé Latin1, est le jeu de caractères par défaut du HTML. Il peut être utilisé pour représenter du texte dans la plupart des langues d'Europe occidentale. Le remplacement de l'ISO-8859-1 par l'ISO-8859-15 comme jeu de caractères HTTP par défaut a été évoqué, car il inclut le nouveau symbole monétaire de l'euro. Cependant, compte tenu de l'adoption généralisée de l'ISO-8859-1, il est peu probable qu'une modification généralisée de l'ISO-8859-15 soit adoptée avant un certain temps.

JIS X 0201

La norme JIS X 0201 est un jeu de caractères extrêmement minimaliste qui étend l'ASCII avec des caractères katakana japonais demi-chasse. Ces caractères étaient initialement utilisés dans le système télégraphique japonais. La norme JIS X 0201 est souvent appelée « JIS Roman ». JIS est l'acronyme de « Japanese Industrial Standard ».

JIS X 0208 et JIS X 0212

Le japonais comprend des milliers de caractères issus de plusieurs systèmes d'écriture. S'il est possible de se débrouiller (péniblement) en utilisant les 63 caractères phonétiques katakana de base de la norme

JIS X 0201, un jeu de caractères beaucoup plus complet est nécessaire pour une utilisation pratique.

Le jeu de caractères JIS X 0208 fut le premier jeu de caractères japonais multi-octets ; il définissait 6 879 caractères codés, dont la plupart sont des kanjis d'origine chinoise.

Le jeu de caractères 0212 ajoute 6 067 caractères supplémentaires.

UCS

Le jeu de caractères universel (UCS) est un effort mondial de normalisation visant à combiner tous les caractères du monde en un seul jeu de caractères codés. L'UCS est défini par la norme ISO 10646. Unicode est un consortium commercial qui assure le suivi des normes UCS. L'UCS offre un espace de codage pour des millions de caractères, bien que le jeu de base ne comporte qu'environ 50 000 caractères.

Schémas de codage des caractères

Les schémas de codage de caractères regroupent les numéros de code de caractère en bits de contenu, puis les décompressent en codes de caractère à l'autre extrémité (figure 16-7). Il existe trois grandes catégories de schémas de codage de caractères :

Largeur fixe

Les codages à largeur fixe représentent chaque caractère codé avec un nombre fixe de bits. Leur traitement est rapide, mais ils peuvent gaspiller de l'espace.

Largeur variable (non modale)

Les codages à largeur variable utilisent un nombre de bits différent pour différents numéros de code de caractère. Ils permettent de réduire le nombre de bits requis pour les caractères courants et conservent la compatibilité avec les anciens jeux de caractères 8 bits tout en permettant l'utilisation de plusieurs octets pour les caractères internationaux.

Largeur variable (modale)

Les codages modaux utilisent des schémas d'échappement spécifiques pour passer d'un mode à l'autre. Par exemple, un codage modal permet de basculer entre plusieurs jeux de caractères se chevauchant au milieu d'un texte. Les codages modaux sont complexes à traiter, mais ils peuvent prendre en charge efficacement des systèmes d'écriture complexes.

Figure 16-7. Le schéma de codage des caractères convertit les codes de caractères en bits et inversement.

Examinons quelques schémas de codage courants.

8 bits

Le codage d'identité à largeur fixe 8 bits code simplement chaque code de caractère avec sa valeur 8 bits correspondante. Il ne prend en charge que les jeux de caractères dont la plage de codes est de 256 caractères. La famille de jeux de caractères ISO-8859 utilise le codage d'identité 8 bits.

UTF-8

UTF-8 est un système d'encodage de caractères populaire conçu pour UCS (UTF signifie « UCS Transformation Format »). UTF-8 utilise un codage non modal à longueur variable pour les valeurs de code de caractère. Les premiers bits du premier octet indiquent la longueur du caractère encodé en octets, et chaque octet suivant contient six bits de valeur de code (voir Tableau 16-2).

Si le premier octet codé possède un bit de poids fort de 0, la longueur est d'un seul octet, les 7 bits restants contenant le code du caractère. Ceci permet d'obtenir une compatibilité ASCII intéressante (mais pas une compatibilité ISO-8859, car ISO-8859 utilise le bit de poids fort).

Tableau 16-2. Codage UTF-8 à largeur variable et non modal

Bits de code de caractère Octet 1 Octet 2 Octet 3 Octet 4 Octet 5 Octet 6

0–7 0ccccccc - - - - -

8–11 110ccccc 10cccccc - - - -

12–16 1110cccc 10cccccc 10cccccc - - -

17–21 11110ccc 10cccccc 10cccccc 10cccccc - -

22–26 111110cc 10cccccc 10cccccc 10cccccc 10cccccc -

27–31 1111110c 10cccccc 10cccccc 10cccccc 10cccccc 10cccccc

Par exemple, le code de caractère 90 (ASCII « Z ») serait codé sur 1 octet (01011010), tandis que le code 5073 (valeur binaire 13 bits 1001111010001) serait codé sur 3 octets :

11100001 10001111 10010001

iso-2022-jp

iso-2022-jp est un codage largement utilisé pour les documents Internet japonais. iso-2022-jp est un codage modal à longueur variable, avec toutes les valeurs inférieures à 128 pour éviter les problèmes avec les logiciels non 8 bits.

Le contexte d'encodage est toujours défini sur l'un des quatre jeux de caractères prédéfinis.* Spécial

Les « séquences d'échappement » passent d'un jeu à un autre. iso-2022-jp utilise initialement le jeu de caractères USASCII, mais il peut passer au jeu de caractères JIS X 0201 (JIS-Roman) ou aux jeux de caractères JIS X 0208-1978 et JIS X 0208-1983 beaucoup plus grands en utilisant des séquences d'échappement de 3 octets.

* L'encodage iso-2022-jp est étroitement lié à ces quatre jeux de caractères, tandis que certains autres encodages sont indépendants du jeu de caractères particulier.

Les séquences d'échappement sont présentées dans le tableau 16-3. En pratique, un texte japonais commence par « ESC $ @ » ou « ESC $ B » et se termine par « ESC (B » ou « ESC (J ».

Tableau 16-3. Séquences d'échappement de changement de jeu de caractères iso-2022-jp

Séquence d'échappement Ensemble de caractères codés résultant Octets par code

ESC (B US-ASCII 1

ESC (J JIS X 0201-1976 (JIS romain) 1

ESC $ @ JIS X 0208-1978 2

ESC $ B JIS X 0208-1983 2

En mode US-ASCII ou JIS-Roman, un seul octet est utilisé par caractère. Avec le jeu de caractères JIS X 0208, plus étendu, deux octets sont utilisés par caractère.

Code. L'encodage limite le nombre d'octets envoyés à 33 et 126.*

euc-jp

euc-jp est un autre codage japonais populaire. EUC signifie « Extended Unix ».

« Code », initialement développé pour prendre en charge les caractères asiatiques sur les systèmes d'exploitation Unix.

Comme l'ISO-2022-JP, l'encodage EUC-JP est un encodage à longueur variable qui permet l'utilisation de plusieurs jeux de caractères japonais standard. Cependant, contrairement à l'ISO-2022-JP, l'encodage EUC-JP n'est pas modal. Il n'y a pas de séquence d'échappement pour passer d'un mode à l'autre.

euc-jp prend en charge quatre jeux de caractères codés : JIS X 0201 (JIS-Roman, ASCII avec quelques substitutions japonaises), JIS X 0208, katakana demi-largeur (63 caractères utilisés dans le système télégraphique japonais d'origine) et JIS X 0212.

Un octet est utilisé pour encoder le JIS Roman (compatible ASCII), deux octets sont utilisés pour le JIS X 0208 et le katakana demi-largeur, et

trois octets sont utilisés pour le JIS X 0212. Le codage est un peu gaspilleur mais est simple à traiter.

Les modèles de codage sont décrits dans le tableau 16-4.

Tableau 16-4. Valeurs de codage euc-jp

Quelles valeurs d'encodage d'octets

JIS X 0201 (94 caractères codés)

1er octet 33–126

JIS X 0208 (6879 caractères codés)

1er octet 161–254

2e octet 161–254

* Bien que les octets ne puissent avoir que 94 valeurs (entre 33 et 126), cela est suffisant pour couvrir tous les caractères des jeux de caractères JIS X 0208, car les jeux de caractères sont organisés dans une grille de 94 × 94 valeurs de code, suffisante pour couvrir tous les codes de caractères JIS X 0208.

Tableau 16-4. Valeurs de codage euc-jp (suite)

Quelles valeurs d'encodage d'octets

Katakana demi-largeur (63 caractères codés)

1er octet 142

2e octet 161–223

JIS X 0212 (6067 caractères codés)

1er octet 143

2e octet 161–254

3e octet 161–254

Ceci conclut notre étude des jeux de caractères et des codages. La section suivante explique les balises de langue et comment HTTP les utilise pour cibler le contenu.

Veuillez vous référer à l’annexe H pour une liste détaillée des jeux de caractères normalisés.

Balises de langue et HTTP

Les balises de langue sont des chaînes courtes et standardisées qui nomment les langues parlées.

Nous avons besoin de noms normalisés, sinon certains utiliseront « French » pour les documents français, d'autres « Français », d'autres

encore « France », et les paresseux utiliseront simplement « Fra » ou « F ». Les étiquettes de langue normalisées évitent cette confusion.

Il existe des balises de langue pour l'anglais (en), l'allemand (de), le coréen (ko) et bien d'autres langues. Ces balises peuvent décrire des variantes régionales et des dialectes, comme le portugais brésilien (pt-BR), l'anglais américain (en-US) et le chinois du Hunan (zhxiang). Il existe même une balise de langue standard pour le klingon (i-klingon) !

L'en-tête de langue de contenu

Le champ d'en-tête d'entité « Contenu-Langue » décrit les langues du public cible de l'entité. Si le contenu est principalement destiné à un public francophone, le champ d'en-tête « Contenu-Langue » contiendra :

Contenu-Langue : fr

L'en-tête Content-Language ne se limite pas aux documents texte. Les clips audio, les films et les applications peuvent tous être destinés à un public linguistique particulier. Tout type de média destiné à un public linguistique particulier peut avoir un en-tête Content-Language. Dans la figure 16-8, le fichier audio est étiqueté pour un public Navajo.

Si le contenu est destiné à plusieurs publics, vous pouvez indiquer plusieurs langues. Comme le suggère la spécification HTTP, une version du « Traité de Waitangi », présentée simultanément en maori et en anglais, nécessiterait :

Contenu-Langue : mi, en

Figure 16-8. L'en-tête Content-Language indique un extrait audio « Rain Song » pour les locuteurs navajos.

Cependant, la présence de plusieurs langues dans une entité ne signifie pas qu'elle est destinée à des publics linguistiques variés. Un manuel d'initiation aux langues, tel que « Une première leçon de latin », clairement destiné à un public anglophone, ne devrait contenir que « en ».

L'en-tête Accept-Language

La plupart d'entre nous connaissons au moins une langue. HTTP nous permet de transmettre nos restrictions et préférences linguistiques aux serveurs web. Si le serveur web possède plusieurs versions d'une langue,

ressource, dans différentes langues, elle peut nous donner du contenu dans notre langue préférée.* Ici, un client demande du contenu en espagnol :

Accepter-Langue : es

Vous pouvez placer plusieurs balises de langue dans l'en-tête Accept-Language pour lister toutes les langues prises en charge et les classer par ordre de préférence (de gauche à droite). Ici, le client préfère l'anglais, mais accepte le suisse allemand (de-CH) ou d'autres variantes de l'allemand (de) :

Langue d'acceptation : en, de-CH, de

Les clients utilisent Accept-Language et Accept-Charset pour demander du contenu qu'ils comprennent. Nous verrons comment cela fonctionne plus en détail au chapitre 17.

Types de balises linguistiques

Les balises de langue ont une syntaxe standardisée, documentée dans la RFC 3066, « Balises pour le

Identification des langues. Les balises de langue peuvent être utilisées pour représenter :

• Cours de langue générale (comme dans « es » pour espagnol)

• Langues spécifiques à un pays (comme dans « en-GB » pour l'anglais en Grande-Bretagne)

• Dialectes des langues (comme dans « no-bok » pour le norvégien « langue des livres »)

* Les serveurs peuvent également utiliser l'en-tête Accept-Language pour générer du contenu dynamique dans la langue de l'utilisateur ou pour sélectionner des images ou des promotions de marchandisage adaptées à la langue cible.

Balises de langue et HTTP

• Langues régionales (comme dans « sgn-US-MA » pour la langue des signes de Martha's Vineyard)

• Langues standardisées non variantes (par exemple, « i-navajo »)

• Langues non standard (par exemple, « x-snowboarder-slang »*)

Sous-étiquettes

Les balises de langue comportent une ou plusieurs parties, séparées par des tirets, appelées sous-balises :

• La première sous-étiquette est appelée sous-étiquette principale. Les valeurs sont standardisées.

• La deuxième sous-balise est facultative et suit sa propre norme de dénomination.

• Toutes les sous-balises de fin ne sont pas enregistrées.

La sous-étiquette principale contient uniquement des lettres (A à Z). Les sous-étiquettes suivantes peuvent contenir des lettres ou des chiffres, jusqu'à huit caractères. Un exemple est présenté à la figure 16-9.

Langue des signes de Martha's Vineyard

sgn-US-MA

Premier sous-balise Deuxième sous-balise Troisième sous-balise

(langue des signes) (Amérique) (Massachusetts)

variante régionale)

Figure 16-9. Les balises de langue sont divisées en sous-balises.

Capitalisation

Toutes les balises sont insensibles à la casse ; les balises « en » et « eN » sont équivalentes. Cependant, les minuscules sont généralement utilisées pour désigner des langues générales, tandis que les majuscules désignent des pays spécifiques. Par exemple, « fr » désigne toutes les langues classées comme françaises, tandis que « FR » désigne la France.†

Enregistrements des balises de langue IANA

Les valeurs des sous-étiquettes de langue première et seconde sont définies par divers documents de normalisation et leurs organismes de maintenance. L'IANA‡ gère la liste des étiquettes de langue standard, conformément aux règles décrites dans la RFC 3066.

Si une balise de langue est composée de valeurs standard de pays et de langue, elle n'a pas besoin d'être enregistrée spécifiquement. Seules les balises de langue qui ne peuvent pas être composées à partir des valeurs standard de pays et de langue doivent être enregistrées spécifiquement auprès de l'autorité de certification.

* Décrit le dialecte unique parlé par les « déchiqueteurs ». † Cette convention est recommandée par la norme ISO 3166.

‡ Voir http://www.iana.org et RFC 2860.

IANA.* Les sections suivantes décrivent les normes RFC 3066 pour les première et deuxième sous-balises.

Première sous-étiquette : espace de noms

La première sous-étiquette est généralement un jeton de langue standardisé, choisi parmi les normes de langage ISO 639. Mais il peut également s'agir de la lettre « i » pour identifier les noms enregistrés

auprès de l'IANA, ou de la lettre « x » pour les noms d'extension privés. Voici les règles :

Si la première sous-étiquette contient :

• Deux caractères, il s'agit d'un code de langue issu des normes ISO 639† et 639-1

• Trois caractères, il s'agit d'un code de langue répertorié dans la norme ISO 639-2‡ et ses extensions

• La lettre « i », la balise de langue est explicitement enregistrée auprès de l'IANA

• La lettre « x », la balise de langue est une sous-balise d'extension privée et non standard

Les noms ISO 639 et 639-2 sont résumés dans l'annexe G. Quelques exemples sont présentés ici dans le tableau 16-5.

Tableau 16-5. Exemples de codes de langue ISO 639 et 639-2

Langue ISO 639 ISO 639-2

Arabe ar ara

Chinois zh chi/zho Néerlandais nl dut/nla

Anglais en eng

Français fr fra/fre

Allemand de deu/ger

Grec (moderne) el ell/gre

hébreu he heb

italien it ita

Japonais ja jpn Coréen ko kor

Norvégien non ni

russe ru rus

Espagnol es esl/spa

* Au moment de la rédaction de ce document, seules 21 balises de langue étaient explicitement enregistrées auprès de l'IANA, dont le cantonais (« zh-yue »), le néo-norvégien (« no-nyn »), le luxembourgeois (« i-lux ») et le klingon (« i-klingon »). Les centaines de langues parlées restantes utilisées sur Internet ont été composées à partir de composants standards.

† Voir la norme ISO 639, « Codes pour la représentation des noms de langues ».

‡ Voir ISO 639-2, « Codes pour la représentation des noms de langues — Partie 2 : Code alpha-3 ».

Balises de langue et HTTP

Tableau 16-5. Exemples de codes de langue ISO 639 et 639-2 (suite)

Langue ISO 639 ISO 639-2

Suédois sv sve/swe

Turc tr tur

Deuxième sous-balise : espace de noms

La deuxième sous-étiquette est généralement un jeton de pays standardisé, choisi parmi les normes ISO 3166 relatives aux codes de pays et aux régions. Il peut également s'agir d'une autre chaîne, que vous pouvez enregistrer auprès de l'IANA. Voici les règles :

Si la deuxième sous-étiquette contient :

• Deux caractères, c'est un pays/une région défini par la norme ISO 3166*

• De trois à huit caractères, il peut être enregistré auprès de l'IANA

• Un seul caractère, c'est illégal

Certains codes pays ISO 3166 sont présentés dans le tableau 16-6. La liste complète des codes pays figure à l'annexe G.

Tableau 16-6. Exemples de codes pays ISO 3166

Code du pays

Brésil BR

Canada CA

Chine CN

France FR

Allemagne DE

Saint-Siège (État de la Cité du Vatican) VA

Hong Kong HK

Inde IN

Italie IT

Japon JP

Liban LB

Mexique MX

Pakistan PK

Fédération de Russie RU

Royaume-Uni GB

États-Unis

* Les codes pays AA, QM–QZ, XA–XZ et ZZ sont réservés par la norme ISO 3166 comme codes attribués par l'utilisateur. Ils ne doivent pas être utilisés pour former des balises de langue.

Sous-balises restantes : espace de noms

Il n'y a pas de règles pour la troisième sous-étiquette et les suivantes, à part le fait qu'elles doivent comporter jusqu'à huit caractères (lettres et chiffres).

Configuration des préférences linguistiques

Vous pouvez configurer les préférences linguistiques dans votre profil de navigateur.

Netscape Navigator vous permet de définir vos préférences linguistiques via Édition → Préférences... → Langues..., et Microsoft Internet Explorer vous permet de définir vos préférences linguistiques via Outils → Options Internet... → Langues.

Tableaux de référence des balises de langue

L'annexe G contient des tableaux de référence pratiques pour les balises de langue :

• Les balises de langue enregistrées par l'IANA sont présentées dans le tableau G-1.

• Les codes de langue ISO 639 sont indiqués dans le tableau G-2.

• Les codes pays ISO 3166 sont indiqués dans le tableau G-3.

URI internationalisés

Aujourd'hui, les URI n'offrent que peu de prise en charge de l'internationalisation. À quelques exceptions près (mal définies), ils sont composés d'un sous-ensemble de caractères US-ASCII. Des efforts sont en cours pour inclure un ensemble plus riche de caractères dans les noms d'hôtes et les chemins d'accès des URL, mais pour l'instant, ces normes ne sont ni largement acceptées ni largement déployées. Passons en revue la pratique actuelle.

Transcriptibilité globale versus caractères significatifs

Les concepteurs d'URI souhaitaient que chacun puisse partager ses URI à travers le monde : par courriel, par téléphone, par panneau d'affichage, et même à la radio. Et ils souhaitaient

Les URI doivent être faciles à utiliser et à mémoriser. Ces deux objectifs sont contradictoires.

Afin de faciliter la saisie, la manipulation et le partage des URI par les utilisateurs du monde entier, les concepteurs ont choisi un ensemble très limité de caractères courants (lettres de l'alphabet latin, chiffres et quelques caractères spéciaux). Ce petit répertoire de caractères est pris en charge par la plupart des logiciels et claviers du monde entier.

Malheureusement, en limitant le jeu de caractères, les concepteurs d'URI ont rendu la création d'URI faciles à utiliser et à mémoriser beaucoup plus difficile pour les utilisateurs du monde entier. La majorité des citoyens du monde ne connaissent même pas l'alphabet latin, ce qui rend quasiment impossible la mémorisation d'URI comme des modèles abstraits.

URI internationalisés

Les auteurs des URI ont estimé qu'il était plus important de garantir la transcripbilité et la partageabilité des identifiants de ressources que de les composer uniquement des caractères les plus significatifs. Ainsi, nous disposons aujourd'hui d'URI essentiellement constitués d'un sous-ensemble restreint de caractères ASCII.

Répertoire de caractères URI

Le sous-ensemble de caractères US-ASCII autorisés dans les URI peut être divisé en classes de caractères réservés, non réservés et d'échappement. Les classes de caractères non réservés peuvent être utilisées de manière générale dans tout composant d'URI qui les autorise. Les caractères réservés ont des significations particulières dans de nombreux URI ; leur utilisation est donc déconseillée. Voir

Tableau 16-7 pour une liste des caractères non réservés, réservés et d'échappement.

Tableau 16-7. Syntaxe des caractères URI

Classe de personnages Répertoire de personnages

Sans réserve [A-Za-z0-9] | «-» | "_" | "." | "!" | « ~ » | «*» | "'" | "(" | ")"

Réservé « ; » | « / » | « ? » | « : » | « @ » | « & » | « = » | « + » | « $ » | « , »

Échapper « % » <HEX> <HEX>

S'échapper et se libérer

Les échappements d'URI permettent d'insérer en toute sécurité des caractères réservés et d'autres caractères non pris en charge (comme les espaces) dans les URI. Un échappement est une séquence de trois caractères, composée d'un pourcentage (%) suivi de deux chiffres

hexadécimaux. Ces deux chiffres hexadécimaux représentent le code d'un caractère US-ASCII.

Par exemple, pour insérer un espace (ASCII 32) dans une URL, vous pouvez utiliser le caractère d'échappement « %20 », car 20 est la représentation hexadécimale de 32. De même, si vous souhaitez inclure un signe de pourcentage et qu'il ne soit pas traité comme un caractère d'échappement, vous pouvez saisir

« %25 », où 25 est la valeur hexadécimale du code ASCII pour le pourcentage.

La figure 16-10 montre comment les caractères conceptuels d'un URI sont convertis en octets de code pour les caractères du jeu de caractères actuel. Lorsque l'URI est requis pour le traitement, les échappements sont annulés, ce qui produit les octets de code ASCII sous-jacents.

En interne, les applications HTTP doivent transporter et transmettre les URI avec les échappements en place. Elles ne doivent déséchappementr les URI que lorsque les données sont nécessaires. Plus important encore, elles doivent s'assurer qu'aucun URI n'est déséchappement deux fois, car les signes de pourcentage éventuellement encodés dans un échappement le seront eux-mêmes, ce qui entraînera une perte de données.

Personnages internationaux en fuite

Notez que les valeurs d'échappement doivent être comprises dans la plage des codes US-ASCII (0-127). Certaines applications tentent d'utiliser des valeurs d'échappement pour représenter les caractères étendus ISO-8859-1 (128-255). Par exemple, les serveurs web peuvent utiliser des valeurs d'échappement pour coder.

Figure 16-10. Les caractères URI sont transportés sous forme d'octets de code échappés, mais traités sans échappement.

Noms de fichiers contenant des caractères internationaux. Ceci est incorrect et peut entraîner des problèmes avec certaines applications.

Par exemple, le nom de fichier Sven Ölssen.html (contenant un tréma) peut être encodé par un serveur web comme suit : Sven%20%D6lssen.html. L'espace peut être encodé avec %20, mais il est techniquement illégal d'encoder le Ö avec %D6, car le code D6 (décimal 214) est hors de portée de l'ASCII. L'ASCII ne définit que les codes jusqu'à 0x7F (décimal 127).

Commutateurs modaux dans les URI

Certains URI utilisent également des séquences de caractères ASCII pour représenter des caractères d'autres jeux de caractères. Par

exemple, le codage iso-2022-jp peut être utilisé pour insérer « ESC (J » pour passer en JIS-Roman et « ESC (B » pour revenir en ASCII). Cela fonctionne dans certaines circonstances locales, mais le comportement n'est pas clairement défini et il n'existe pas de schéma normalisé pour identifier le codage spécifique utilisé pour l'URL. Comme le précisent les auteurs de la RFC 2396 :

En revanche, pour les séquences de caractères originales contenant des caractères non ASCII, la situation est plus complexe. Les protocoles Internet qui transmettent des séquences d'octets destinées à représenter des séquences de caractères doivent fournir un moyen d'identifier le jeu de caractères utilisé, s'il y en a plusieurs [RFC2277].

Cependant, la syntaxe générique des URI ne prévoit actuellement aucune disposition permettant cette identification. Un schéma d'URI individuel peut exiger un jeu de caractères unique, définir un jeu de caractères par défaut ou fournir un moyen d'indiquer le jeu de caractères utilisé. Un traitement systématique du codage des caractères dans les URI devrait être développé lors d'une future modification de cette spécification.

Actuellement, les URI ne sont pas très compatibles avec l'international. L'objectif de portabilité des URI a pris le pas sur celui de flexibilité linguistique. Des efforts sont actuellement déployés pour internationaliser les URI, mais à court terme, les applications HTTP devraient s'en tenir à cette norme.

L'ASCII existe depuis 1968, donc ce n'est pas si mal.

URI internationalisés

Autres considérations

Cette section aborde quelques autres éléments que vous devez garder à l’esprit lors de l’écriture d’applications HTTP internationales.

En-têtes et données hors spécifications

Les en-têtes HTTP doivent contenir des caractères du jeu de caractères US-ASCII. Cependant, tous les clients et serveurs ne l'implémentent pas correctement ; vous pouvez donc parfois recevoir des caractères illégaux dont les valeurs de code sont supérieures à 127.

De nombreuses applications HTTP utilisent des routines de système d'exploitation et de bibliothèque pour traiter les caractères (par exemple, la bibliothèque de classification de caractères Unix ctype). Toutes ces bibliothèques ne prennent pas en charge les codes de caractères hors de la plage ASCII (0-127).

Dans certains cas (généralement avec les implémentations plus anciennes), ces bibliothèques peuvent renvoyer des résultats incorrects ou faire planter l'application lorsqu'elles reçoivent des caractères non ASCII. Lisez attentivement la documentation de vos bibliothèques de

classification de caractères avant de les utiliser pour traiter des messages HTTP, au cas où ceux-ci contiendraient des données illégales.

Dates

La spécification HTTP définit clairement les formats de date GMT légaux, mais sachez que tous les serveurs et clients web ne respectent pas ces règles. Par exemple, nous avons constaté que des serveurs web envoyaient des en-têtes de date HTTP non valides avec des mois exprimés dans les langues locales.

Les applications HTTP doivent s'efforcer de tolérer les dates hors spécifications et de ne pas planter à la réception, mais elles ne sont pas toujours capables d'interpréter toutes les dates envoyées. Si la date n'est pas analysable, les serveurs doivent la traiter avec prudence.

Noms de domaine

Le DNS ne prend actuellement pas en charge les caractères internationaux dans les noms de domaine. Des travaux de normalisation sont en cours pour prendre en charge les noms de domaine multilingues, mais ils ne sont pas encore largement déployés.

Pour plus d'informations

Le succès du World Wide Web implique que les applications HTTP continueront d'échanger de plus en plus de contenu dans différentes langues et jeux de caractères. Pour plus d'informations sur le sujet important, mais quelque peu complexe, du multimédia multilingue, veuillez consulter les sources suivantes.

Annexes

• Les balises de jeu de caractères enregistrées par l'IANA sont répertoriées dans le tableau H-1.

• Les balises de langue enregistrées par l’IANA sont présentées dans le tableau G-1.

• Les codes de langue ISO 639 sont indiqués dans le tableau G-2.

• Les codes pays ISO 3166 sont indiqués dans le tableau G-3.

Internationalisation d'Internet

http://www.w3.org/International/

« Rendre le WWW véritablement mondial » — l'internationalisation et

Site Web de localisation.

http://www.ietf.org/rfc/rfc2396.txt

La RFC 2396, « Uniform Resource Identifiers (URI) : Syntaxe générique », définit les URI. Ce document comprend des sections

décrivant les restrictions de jeu de caractères pour les URI internationaux.

Traitement de l'information CJKV

Ken Lunde, O'Reilly & Associates, Inc. CJKV est la bible du traitement électronique des caractères asiatiques. Les jeux de caractères asiatiques sont variés et complexes, mais cet ouvrage offre une excellente introduction aux technologies standard pour les jeux de caractères volumineux.

http://www.ietf.org/rfc/rfc2277.txt

La RFC 2277, « Politique de l'IETF sur les jeux de caractères et les langues », documente les politiques actuelles appliquées par l'Internet Engineering Steering Group (IESG) aux efforts de normalisation de l'Internet Engineering Task Force (IETF) afin d'aider les protocoles Internet à échanger des données dans plusieurs langues et

personnages.

Normes internationales

http://www.iana.org/numbers.htm

L'Autorité d'assignation des numéros sur Internet (IANA) gère des répertoires de noms et de numéros enregistrés. Le « Répertoire des numéros et attributions de protocole » répertorie les jeux de caractères enregistrés destinés à être utilisés sur Internet. Étant donné qu'une grande partie des travaux sur les communications internationales relèvent de la compétence de l'ISO, et non de la communauté Internet, les listes de l'IANA ne sont pas exhaustives.

http://www.ietf.org/rfc/rfc3066.txt

La RFC 3066, « Balises pour l’identification des langues », décrit les balises de langue, leurs valeurs et comment les construire.

Pour plus d'informations

« Codes pour la représentation des noms de langues »

ISO 639:1988 (E/F), Organisation internationale de normalisation, première édition.

« Codes pour la représentation des noms de langues — Partie 2 : Code alpha-3 »

ISO 639-2:1998, Groupe de travail mixte de l'ISO TC46/SC4 et de l'ISO TC37/SC2, première édition.

« Codes pour la représentation des noms de pays »

ISO 3166:1988 (E/F), Organisation internationale de normalisation, troisième édition.

Chapitre 17Ceci est le titre du livre CHAPITRE 17

Négociation de contenu et transcodage

Souvent, une même URL doit correspondre à différentes ressources. Prenons l'exemple d'un site web souhaitant proposer son contenu en plusieurs langues. Si un site comme Joe's Hardware compte des utilisateurs francophones et anglophones, il pourrait être judicieux de proposer son site web dans les deux langues. Cependant, si un client de Joe demande « http://www.joes-hardware.com », quelle version le serveur doit-il envoyer ? Français ou anglais ?

Idéalement, le serveur enverra la version anglaise à un anglophone et la version française à un francophone. Un utilisateur pourrait ainsi accéder à la page d'accueil de Joe's Hardware et obtenir du contenu dans sa langue. Heureusement, HTTP offre des méthodes de négociation de contenu qui permettent aux clients et aux serveurs de faire exactement cela. Grâce à ces méthodes, une même URL peut correspondre à différentes ressources (par exemple, une ressource française et une ressource française).

(version anglaise de la même page web). Ces différentes versions sont appelées variantes.

Les serveurs peuvent également prendre d'autres décisions concernant le contenu le plus pertinent à envoyer à un client pour une URL donnée. Dans certains cas, ils peuvent même générer automatiquement des pages personnalisées ; par exemple, un serveur peut convertir une page HTML en page WML pour votre appareil mobile. Ces transformations de contenu dynamiques sont appelées transcodages. Elles sont effectuées en réponse à la négociation de contenu entre les clients HTTP et les serveurs.

Dans ce chapitre, nous discuterons de la négociation de contenu et de la manière dont les applications Web s'acquittent de leurs tâches de négociation de contenu.

Techniques de négociation de contenu

Il existe trois méthodes distinctes pour déterminer quelle page d'un serveur est la plus adaptée à un client : présenter le choix au client, décider automatiquement sur le serveur ou demander à un intermédiaire de sélectionner. Ces trois techniques sont respectivement appelées négociation pilotée par le client, négociation pilotée par le serveur et négociation transparente (voir tableau 17-1).

Dans ce chapitre, nous examinerons les mécanismes de chaque technique ainsi que leurs avantages et inconvénients.

Tableau 17-1. Résumé des techniques de négociation de contenu

Technique	Comment ça marche	Avantages	Inconvénients
Piloté par le client	le client effectue une requête, le serveur lui envoie une liste de choix, et le client choisit.	La mise en œuvre est plus simple côté serveur.	Le client peut faire le meilleur choix.	Latence accrue : au moins deux requêtes sont nécessaires pour obtenir le contenu correct.
Piloté par le serveur	le serveur examine les en-têtes de requête du client et décide de la version à servir.	Plus rapide que la négociation pilotée par le client. HTTP fournit un mécanisme de valeur Q permettant aux serveurs d'établir des correspondances approximatives, ainsi qu'un en-tête Vary permettant aux serveurs d'indiquer aux périphériques en aval comment évaluer les requêtes.	Si la décision n'est pas évidente (en-têtes non concordants), le serveur doit deviner.
Transparent	un périphérique intermédiaire (généralement un cache proxy) négocie la requête pour le compte du client.	Décharge la négociation du serveur Web. Plus rapide que la négociation pilotée par le client.	Aucune spécification formelle sur la manière de réaliser une négociation transparente.

Négociation axée sur le client

La solution la plus simple pour un serveur lorsqu'il reçoit une requête client est de renvoyer une réponse listant les pages disponibles et de laisser le client choisir celle qu'il souhaite consulter. C'est bien sûr la solution la plus simple à mettre en œuvre sur le serveur et elle permet généralement de sélectionner la meilleure copie (à condition que la liste contienne suffisamment d'informations pour permettre au client de choisir la bonne copie). L'inconvénient est que deux requêtes sont nécessaires pour chaque page : une pour obtenir la liste et une seconde pour obtenir la copie sélectionnée. Ce processus est lent et fastidieux, et risque d'être agaçant pour le client.

Mécaniquement, les serveurs disposent de deux méthodes pour présenter les choix au client : en renvoyant un document HTML contenant des liens vers les différentes versions de la page et leur description, ou en renvoyant une réponse HTTP/1.1 avec le code de réponse 300 Choix multiples. Le navigateur client peut recevoir cette réponse et afficher une page contenant les liens, comme dans la première méthode, ou afficher une boîte de dialogue invitant l'utilisateur à faire une sélection. Dans tous les cas, la décision est prise manuellement côté client par l'utilisateur du navigateur.

Outre la latence accrue et la gêne occasionnée par les multiples requêtes par page, cette méthode présente un autre inconvénient : elle nécessite plusieurs URL : une pour la page principale et une pour chaque page spécifique. Ainsi, si la requête initiale concernait

www.joeshardware.com, le serveur de Joe pourrait renvoyer une page contenant des liens vers www.joeshardware.com/english et www.joes-hardware.com/french. Les clients doivent-ils désormais ajouter à leurs favoris la page principale d'origine ou les pages sélectionnées ? Doivent-ils parler de l'excellent site web www.joes-hardware.com à leurs amis ou ne parler que de celui www.joes-hardware.com/english à leurs amis anglophones ?

Négociation pilotée par le serveur

La négociation pilotée par le client présente plusieurs inconvénients, comme évoqué dans la section précédente. La plupart de ces inconvénients concernent la communication accrue entre le client et le serveur pour déterminer la page la plus adaptée à une requête. Une solution pour réduire cette communication supplémentaire consiste à laisser le serveur décider de la page à renvoyer. Pour ce faire, le client doit fournir suffisamment d'informations sur ses préférences pour permettre au serveur de prendre une décision éclairée. Le serveur obtient ces informations des en-têtes de requête du client.

Les serveurs HTTP utilisent deux mécanismes pour évaluer la réponse appropriée à envoyer à un client :

• Examen de l'ensemble des en-têtes de négociation de contenu. Le serveur examine les en-têtes « Accept » du client et tente de les associer aux en-têtes de réponse correspondants.

• Variation selon d'autres en-têtes (hors négociation de contenu). Par exemple, le serveur pourrait envoyer des réponses basées sur l'en-tête User-Agent du client.

Ces deux mécanismes sont expliqués plus en détail dans les sections suivantes.

En-têtes de négociation de contenu

Les clients peuvent envoyer leurs informations de préférence à l’aide de l’ensemble d’en-têtes HTTP répertoriés dans le tableau 17-2.

Tableau 17-2. Accepter les en-têtes

Description de l'en-tête

Accepter Utilisé pour indiquer au serveur quels types de médias peuvent être envoyés

Accept-Language Utilisé pour indiquer au serveur quelles langues peuvent être envoyées

Accept-Charset Utilisé pour indiquer au serveur quels jeux de caractères peuvent être envoyés

Accept-Encoding Utilisé pour indiquer au serveur quels encodages peuvent être envoyés

Notez la similitude de ces en-têtes avec les en-têtes d'entité décrits au chapitre 15. Cependant, il existe une distinction claire entre les objectifs des deux types d'en-têtes. Comme mentionné au chapitre 15, les en-têtes d'entité sont comparables à des étiquettes d'expédition : ils spécifient les attributs du corps du message nécessaires au transfert des messages du serveur au client. Les en-têtes de négociation de contenu, quant à eux, sont utilisés par les clients et les serveurs pour échanger des informations de préférences et faciliter le choix entre différentes versions d'un document, afin que celle qui correspond le mieux aux préférences du client soit servie.

Les serveurs font correspondre les en-têtes Accept des clients avec les en-têtes d'entité correspondants, répertoriés dans le tableau 17-3.

Négociation pilotée par le serveur

Tableau 17-3. Accepter et associer les en-têtes de documents

Accepter l'en-tête Entité en-tête

Accepter le type de contenu

Accepter-Langue Contenu-Langue

Accepter-Charset Content-Type

Accepter-Encodage Contenu-Encodage

Notez que comme HTTP est un protocole sans état (ce qui signifie que les serveurs ne gardent pas trace des préférences des clients entre les requêtes), les clients doivent envoyer leurs informations de préférence avec chaque requête.

Si les deux clients envoient l'en-tête Accept-Language spécifiant la langue qui les intéresse, le serveur peut choisir la copie de www.joes-hardware.com à renvoyer à chaque client. Laisser le serveur choisir automatiquement le document à renvoyer réduit la latence liée aux échanges requis par le modèle client.

Supposons qu'un client préfère l'espagnol. Quelle version de la page le serveur doit-il renvoyer ? L'anglais ou le français ? Le serveur n'a que deux choix : deviner, ou se rabattre sur le modèle client et lui demander de choisir. Cependant, si l'Espagnol comprend un peu l'anglais, il pourrait choisir la page en anglais ; ce ne serait pas idéal, mais cela suffirait. Dans ce cas, l'Espagnol doit pouvoir communiquer davantage d'informations sur ses préférences, en faisant comprendre qu'il a une connaissance minimale de l'anglais et qu'en cas de besoin, l'anglais suffira.

Heureusement, HTTP fournit un mécanisme permettant aux clients comme notre Espagnol de donner des descriptions plus riches de leurs préférences, en utilisant des valeurs de qualité (« valeurs q » en abrégé).

Valeurs de qualité de l'en-tête de négociation de contenu

Le protocole HTTP définit des valeurs de qualité permettant aux clients de lister plusieurs choix pour chaque catégorie de préférence et d'associer un ordre de préférence à chaque choix. Par exemple, les clients peuvent envoyer un en-tête Accept-Language du formulaire suivant :

Accepter-Langue : en ; q = 0,5, fr ; q = 0,0, nl ; q = 1,0, tr ; q = 0,0

Les valeurs q peuvent varier de 0,0 à 1,0 (0,0 étant la préférence la plus basse et 1,0 la plus élevée). L'en-tête ci-dessus indique donc que le client préfère recevoir une version néerlandaise (nl) du document, mais qu'une version anglaise (en) fera l'affaire. En revanche, le client ne souhaite en aucun cas une version française (fr) ou turque (tr). Notez que l'ordre d'affichage des préférences n'a pas d'importance ; seules les valeurs q qui leur sont associées le sont.

Il arrive que le serveur ne dispose d'aucun document correspondant aux préférences du client. Dans ce cas, le serveur peut modifier ou transcoder le document pour qu'il corresponde aux préférences du client. Ce mécanisme est abordé plus loin dans ce chapitre.

Variant sur d'autres en-têtes

Les serveurs peuvent également tenter de faire correspondre les réponses avec d'autres en-têtes de requête client, tels que User-Agent. Par exemple, les serveurs peuvent savoir que les anciennes versions d'un navigateur ne prennent pas en charge JavaScript et peuvent donc renvoyer une version de la page dépourvue de JavaScript.

Dans ce cas, il n'existe aucun mécanisme de valeur q pour rechercher les correspondances approximatives les plus « optimales ». Le serveur recherche une correspondance exacte ou diffuse simplement ce qu'il a (selon son implémentation).

Comme les caches doivent tenter de fournir les meilleures versions des documents mis en cache, le protocole HTTP définit un en-tête Vary que le serveur envoie dans les réponses. Cet en-tête indique aux caches (et aux clients, ainsi qu'aux proxys en aval) les en-têtes utilisés par le serveur pour déterminer la meilleure version de la réponse à envoyer. L'en-tête Vary est abordé plus en détail plus loin dans ce chapitre.

Négociation de contenu sur Apache

Voici un aperçu de la façon dont le serveur web Apache prend en charge la négociation de contenu. Il appartient au fournisseur de contenu du site web, Joe par exemple, de fournir différentes versions de sa page d'index. Joe doit placer tous ses fichiers de page d'index dans le répertoire approprié du serveur Apache correspondant à son site web. Il existe deux façons d'activer la négociation de contenu :

• Dans le répertoire du site web, créez un fichier de mappage de types pour chaque URI du site web comportant des variantes. Ce fichier répertorie toutes les variantes et les en-têtes de négociation de contenu auxquels elles correspondent.

• Activez la directive MultiViews, qui oblige Apache à créer automatiquement des fichiers de mappage de types pour le répertoire.

Utilisation de fichiers de mappage de types

Le serveur Apache doit connaître l'apparence des fichiers de correspondance de types. Pour ce faire, définissez un gestionnaire dans le fichier de configuration du serveur qui spécifie le suffixe des fichiers de correspondance de types. Par exemple :

AddHandler type-map .var

Cette ligne indique que les fichiers avec l'extension .var sont des fichiers de mappage de type.

Voici un exemple de fichier de mappage de types : URI : joes-hardware.html

URI : joes-hardware.fr.html

Type de contenu : texte/html

Langue du contenu : en

Négociation pilotée par le serveur

URI : joes-hardware.fr.de.html

Type de contenu : texte/html ; charset = iso-8859-2

Contenu-langue : fr, de

Grâce à ce fichier de correspondance de types, le serveur Apache sait qu'il doit envoyer joes-hardware.en.html aux clients demandant de l'anglais et joes-hardware.fr.de.html aux clients demandant du français. Les valeurs de qualité sont également prises en charge ; consultez la documentation du serveur Apache.

Utilisation de MultiViews

Pour utiliser MultiViews, vous devez l'activer pour le répertoire contenant le site Web, à l'aide d'une directive Options dans la section appropriée du fichier access.conf (<Directory>, <Location> ou <Files>).

Si MultiViews est activé et qu'un navigateur demande une ressource nommée joes-hardware, le serveur recherche tous les fichiers dont le nom contient « joes-hardware » et crée un fichier de correspondance de types pour chacun d'eux. En fonction de ces noms, le serveur détermine les en-têtes de négociation de contenu correspondants. Par exemple, une version française de joes-hardware devrait contenir l'extension .fr.

Extensions côté serveur

Une autre façon d'implémenter la négociation de contenu sur le serveur consiste à utiliser des extensions côté serveur, telles que les pages ASP (Active Server Pages) de Microsoft. Consultez le chapitre 8 pour un aperçu des extensions côté serveur.

Négociation transparente

La négociation transparente vise à décharger le serveur de la charge de négociation, tout en minimisant les échanges de messages avec le client, grâce à un proxy intermédiaire qui négocie pour le compte du client. Le proxy est supposé connaître les attentes du client et être capable de mener les négociations en son nom (il a reçu les attentes du client dans la demande de contenu). Pour prendre en charge la négociation transparente de contenu, le serveur doit pouvoir indiquer aux proxys les en-têtes de requête qu'il examine pour déterminer la meilleure correspondance avec la requête du client. La spécification HTTP/1.1 ne définit aucun mécanisme de négociation transparente, mais définit l'en-tête Vary. Les serveurs envoient des en-têtes Vary dans leurs réponses pour indiquer aux intermédiaires les en-têtes de requête qu'ils utilisent pour le contenu.

négociation.

Les proxys de mise en cache peuvent stocker différentes copies de documents accessibles via une même URL. Si les serveurs communiquent leurs processus décisionnels aux caches, ces derniers peuvent négocier avec les clients pour le compte des serveurs. Les caches sont également d'excellents outils pour transcoder du contenu, car un transcodeur polyvalent déployé dans un cache peut transcoder le contenu de n'importe quel serveur, et non d'un seul. Le transcodage de contenu dans un cache est illustré à la figure 17-3 et détaillé plus loin dans ce chapitre.

Mise en cache et alternatives

La mise en cache du contenu suppose sa réutilisation ultérieure. Cependant, les caches doivent utiliser une grande partie de la logique décisionnelle utilisée par les serveurs lors de l'envoi d'une réponse, afin de garantir la bonne réponse mise en cache à la requête client.

La section précédente décrit les en-têtes « Accept » envoyés par les clients et les en-têtes d'entité correspondants que les serveurs comparent afin de choisir la meilleure réponse à chaque requête. Les caches doivent utiliser ces mêmes en-têtes pour déterminer la réponse mise en cache à renvoyer.

La figure 17-1 illustre une séquence d'opérations correcte et incorrecte impliquant un cache. La première requête est transmise au serveur par le cache et la réponse est stockée. La seconde réponse est recherchée par le cache et un document correspondant à l'URL est trouvé. Or, ce

document est en français, et le demandeur souhaite un document en espagnol. Si le cache renvoie uniquement le document en français au demandeur, son comportement sera incorrect.

Figure 17-1. Les caches utilisent des en-têtes de négociation de contenu pour renvoyer des réponses correctes aux clients.

Le cache doit donc également transmettre la seconde requête au serveur et stocker la réponse ainsi qu'une réponse « alternative » pour cette URL. Le cache contient désormais deux

Négociation transparente

Différents documents pour une même URL, comme le fait le serveur. Ces différentes versions sont appelées variantes ou alternatives. La négociation de contenu peut être considérée comme le processus de sélection, parmi les variantes, de la meilleure correspondance à la requête d'un client.

L'en-tête Vary

Voici un ensemble typique d'en-têtes de demande et de réponse provenant d'un navigateur et d'un serveur :

OBTENIR http://www.joes-hardware.com/ HTTP/1.0

Connexion proxy : Keep-Alive

Agent utilisateur : Mozilla/4.73 [fr] (WinNT; U)

Hébergeur : www.joes-hardware.com

Accepter : image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*

Accepter-Encodage : gzip

Accepter-Langue : en, pdf

Accept-Charset : iso-8859-1, *, utf-8

HTTP/1.1 200 OK

Date : dim. 10 déc. 2000 22:13:40 GMT

Serveur : Apache/1.3.12 OpenSSL/0.9.5a (Unix) FrontPage/4.0.4.3

Dernière modification : ven. 5 mai 2000 04:42:52 GMT

Étiquette : « 1b7ddf-48-3912514c »

Accept-Ranges : octets

Contenu-Longueur : 72

Connexion : fermer

Type de contenu : texte/html

Que se passe-t-il, cependant, si la décision du serveur repose sur des en-têtes autres que les en-têtes Accept, comme l'en-tête User-Agent ? Ce n'est pas aussi radical qu'il y paraît. Les serveurs peuvent savoir que les anciennes versions d'un navigateur ne prennent pas en charge JavaScript, par exemple, et peuvent donc renvoyer une version de la page dépourvue de JavaScript. Si les serveurs utilisent d'autres en-têtes pour décider quelles pages renvoyer, les caches doivent connaître ces en-têtes afin de pouvoir appliquer une logique parallèle pour choisir la page en cache à renvoyer.

L'en-tête de réponse HTTP Vary répertorie tous les en-têtes de requête client pris en compte par le serveur pour sélectionner le document ou générer du contenu personnalisé (en plus des en-têtes de négociation de contenu habituels). Par exemple, si le document servi dépend de l'en-tête User-Agent, l'en-tête Vary doit inclure « User-Agent ».

Lorsqu'une nouvelle requête arrive, le cache trouve la meilleure correspondance grâce aux en-têtes de négociation de contenu. Avant de pouvoir servir ce document au client, il doit vérifier si le serveur a envoyé un en-tête Vary dans la réponse mise en cache. Si un en-tête Vary est présent, les valeurs des en-têtes de la nouvelle requête doivent correspondre à celles de l'ancienne requête mise en cache. Les serveurs pouvant adapter leurs réponses en fonction des en-têtes des requêtes client, les caches doivent stocker à la fois les en-têtes des requêtes client et les en-têtes des réponses serveur correspondantes avec chaque variable mise en cache, afin de mettre en œuvre une négociation transparente. Ceci est illustré à la figure 17-2.

Figure 17-2. Si les serveurs diffèrent sur certains en-têtes de requête, les caches doivent correspondre à ces en-têtes, en plus des en-têtes de négociation de contenu habituels, avant de renvoyer les réponses mises en cache.

Si l'en-tête Vary d'un serveur ressemblait à ceci, le grand nombre de valeurs User-Agent et Cookie différentes pourrait générer de nombreuses variantes :

Varier : agent utilisateur, cookie

Un cache doit stocker chaque version de document correspondant à chaque variante. Lors d'une recherche, le cache effectue d'abord une correspondance de contenu avec les en-têtes de négociation de contenu, puis compare la variante de la requête avec les variantes mises en cache. En l'absence de correspondance, le cache récupère le document depuis le serveur d'origine.

Transcodage

Nous avons discuté en détail du mécanisme par lequel les clients et les serveurs peuvent choisir entre un ensemble de documents pour une URL et envoyer celui qui correspond le mieux à l'URL.

Transcodage

Les besoins du client. Ces mécanismes reposent sur la présence de documents correspondant aux besoins du client, qu'ils soient parfaitement ou non.

Mais que se passe-t-il lorsqu'un serveur ne dispose d'aucun document correspondant aux besoins du client ? Le serveur peut être amené à renvoyer une erreur, mais théoriquement, il peut être capable de transformer l'un de ses documents existants en un document exploitable par le client. Cette option est appelée transcodage.

Le tableau 17-4 répertorie quelques transcodages hypothétiques.

Tableau 17-4. Transcodages hypothétiques

Avant Après

Document HTML Document WML

Image haute résolution Image basse résolution

Image en 64 000 couleurs Image en noir et blanc

Page complexe avec cadres Page de texte simple sans cadres ni images

Page HTML avec applets Java Page HTML sans applets Java

Page avec publicités Page avec publicités supprimées

Il existe trois catégories de transcodage : la conversion de format, la synthèse d’informations et l’injection de contenu.

Conversion de format

La conversion de format est la transformation de données d'un format à un autre afin de les rendre lisibles par un client. Un appareil sans fil cherchant à accéder à un document généralement visualisé par un client de bureau peut le faire grâce à une conversion HTML vers WML. Un client accédant à une page web via une liaison lente et peu intéressé par les images haute résolution peut visualiser plus facilement une page riche en images si la taille et la résolution des images sont réduites, en les convertissant de la couleur au noir et blanc et en les réduisant.

La conversion de format est pilotée par les en-têtes de négociation de contenu répertoriés dans le tableau 17-2, mais peut également être pilotée par l'en-tête User-Agent. Notez que la transformation de contenu, ou transcodage, diffère de l'encodage de contenu ou de l'encodage de transfert, car ces deux derniers sont généralement

utilisés pour un transport plus efficace et plus sûr du contenu, tandis que le premier sert à le rendre visible sur le périphérique d'accès.

Synthèse de l'information

L'extraction d'informations clés d'un document (appelée synthèse d'information) peut constituer un processus de transcodage utile. Un exemple simple est la génération du plan d'un document à partir des titres de section, ou la suppression de publicités et de logos d'une page.

Des technologies plus sophistiquées, qui catégorisent les pages en fonction de mots-clés, sont également utiles pour résumer l'essentiel d'un document. Cette technologie est souvent utilisée par les systèmes de classification automatique des pages web, tels que les annuaires de pages web des portails.

Injection de contenu

Les deux catégories de transcodages décrites jusqu'à présent réduisent généralement la quantité de contenu des documents web, mais il existe une autre catégorie de transformations qui l'augmente : les transcodages par injection de contenu. Les générateurs automatiques de publicités et les systèmes de suivi des utilisateurs en sont des exemples.

Imaginez l'attrait (et l'offense) d'un transcodeur d'insertion publicitaire qui ajoute automatiquement des publicités à chaque page HTML au fur et à mesure de son affichage. Ce type de transcodage doit être dynamique : il doit être effectué à la volée pour être efficace et ajouter des publicités pertinentes ou ciblées pour un utilisateur spécifique. Des systèmes de suivi des utilisateurs peuvent également être conçus pour ajouter du contenu aux pages de manière dynamique, afin de recueillir des statistiques sur la consultation de la page et la navigation des clients sur le Web.

Transcodage versus prégénération statique

Une alternative au transcodage consiste à créer différentes copies des pages web sur le serveur : par exemple, une avec HTML, une avec WML, une avec des images haute résolution, une avec des images basse résolution, une avec du contenu multimédia et une sans. Cette technique est toutefois peu pratique, et ce pour plusieurs raisons : la moindre modification d'une page nécessite la modification de plusieurs pages, l'espace de stockage des différentes versions de chaque page est plus important, et il est plus difficile de cataloguer les pages et de programmer les serveurs web pour qu'ils les utilisent correctement. Certains transcodages, comme l'insertion de publicités (notamment ciblées), ne peuvent pas être effectués de manière statique : l'insertion de la publicité dépend de l'utilisateur qui demande la page.

La transformation à la volée d'une seule page racine peut être une solution plus simple que la prégénération statique. Cependant, elle peut entraîner une latence accrue lors de la diffusion du contenu. Une partie de ces calculs peut toutefois être effectuée par un tiers, déchargeant ainsi le serveur web de la tâche de traitement. La transformation peut être effectuée par un agent externe sur un proxy ou un cache. La figure 17-3 illustre le transcodage sur un cache proxy.

Prochaines étapes

L'histoire de la négociation de contenu ne s'arrête pas avec les en-têtes Accept et Content, pour plusieurs raisons :

• La négociation de contenu en HTTP entraîne des limitations de performances. Rechercher un contenu approprié parmi de nombreuses variantes, ou essayer de deviner la meilleure correspondance, peut

Prochaines étapes

Figure 17-3. Transformation ou transcodage de contenu dans un cache proxy

être coûteux. Existe-t-il des moyens de rationaliser et de cibler le protocole de négociation de contenu ? Les RFC 2295 et 2296 tentent de répondre à cette question pour une négociation de contenu HTTP transparente.

HTTP n'est pas le seul protocole nécessitant une négociation de contenu. Le streaming multimédia et le fax sont deux autres exemples où client et serveur doivent discuter de la meilleure réponse à la requête du client. Un protocole général de négociation de contenu peut-il être développé sur la base des protocoles d'application TCP/IP ? Le groupe de travail sur la négociation de contenu a été créé pour répondre à cette question. Ce groupe est désormais fermé, mais il a contribué à plusieurs RFC. Consultez la section suivante pour accéder au site web du groupe.

Pour plus d'informations

Les projets Internet et la documentation en ligne suivants peuvent vous donner plus de détails sur la négociation de contenu :

http://www.ietf.org/rfc/rfc2616.txt

La RFC 2616, « Hypertext Transfer Protocol—HTTP/1.1 », est la spécification officielle de HTTP/1.1, la version actuelle du protocole HTTP. Bien rédigée, bien structurée et détaillée, cette spécification n'est toutefois pas idéale pour les lecteurs souhaitant comprendre les concepts et les motivations sous-jacents de HTTP, ni les différences entre théorie et pratique. Nous espérons que ce livre vous éclairera sur

les concepts sous-jacents afin que vous puissiez mieux exploiter la spécification. http://search.ietf.org/rfc/rfc2295.txt

La RFC 2295, « Transparent Content Negotiation in HTTP », décrit un protocole de négociation de contenu transparent basé sur HTTP. Ce protocole reste expérimental.

http://search.ietf.org/rfc/rfc2296.txt

La RFC 2296, « Algorithme de sélection de variantes à distance HTTP — RVSA 1.0 », décrit un algorithme permettant de sélectionner de manière transparente le « meilleur » contenu pour une requête HTTP donnée. Ce document reste expérimental.

http://search.ietf.org/rfc/rfc2936.txt

La RFC 2936, « Détection des gestionnaires de types MIME HTTP », décrit une approche permettant de déterminer les gestionnaires de types MIME pris en charge par un navigateur. Cette approche peut s'avérer utile si l'en-tête Accept n'est pas suffisamment précis.

http://www.imc.org/ietf-medfree/index.htm

Ceci est un lien vers le groupe de travail sur la négociation de contenu (CONNEG), qui s'est penché sur la négociation de contenu transparente pour HTTP, le fax et l'impression. Ce groupe est désormais fermé.

Pour plus d'informations

PARTIE V

V. Publication et distribution de contenu

La partie V parle de la technologie de publication et de diffusion de contenu Web :

• Le chapitre 18, Hébergement Web, décrit les façons dont les gens déploient des serveurs dans des environnements d'hébergement Web modernes, la prise en charge HTTP pour l'hébergement Web virtuel et la manière de répliquer du contenu sur des serveurs géographiquement distants.

• Le chapitre 19, Systèmes de publication, traite des technologies permettant de créer du contenu Web et de l’installer sur des serveurs Web.

• Le chapitre 20, Redirection et équilibrage de charge, examine les outils et techniques de distribution du trafic Web entrant entre un ensemble de serveurs.

• Le chapitre 21, Journalisation et suivi de l'utilisation, couvre les formats de journaux et les

des questions.

Chapitre 18Ceci est le titre du livre CHAPITRE 18

Hébergement Web

Lorsque vous placez des ressources sur un serveur web public, vous les mettez à disposition de la communauté Internet. Ces ressources peuvent être aussi simples que des fichiers texte ou des images, ou aussi complexes que des cartes routières en temps réel ou des plateformes de commerce électronique. Il est essentiel que cette riche variété de ressources, détenues par différentes organisations, puisse être facilement publiée sur des sites web et hébergée sur des serveurs offrant de bonnes performances à un prix raisonnable.

L'hébergement web regroupe les fonctions de stockage, de courtage et d'administration des ressources de contenu. L'hébergement est l'une des principales fonctions d'un serveur web. Vous avez besoin d'un serveur pour héberger, diffuser, enregistrer les accès et administrer votre contenu. Si vous ne souhaitez pas gérer vous-même le matériel et les logiciels nécessaires, vous avez besoin d'un hébergeur. Les hébergeurs louent des services de diffusion et d'administration de sites web et offrent différents niveaux de sécurité, de reporting et de simplicité d'utilisation. Ils regroupent généralement les sites web sur des serveurs performants pour des raisons de rentabilité, de fiabilité et de performances.

Ce chapitre explique certaines des fonctionnalités les plus importantes des services d'hébergement web et leur interaction avec les applications HTTP. Il aborde notamment :

• Comment différents sites Web peuvent être « hébergés virtuellement » sur le même serveur et comment cela affecte HTTP

• Comment rendre les sites Web plus fiables en cas de trafic intense

• Comment accélérer le chargement des sites Web

Services d'hébergement

Aux débuts du World Wide Web, les organisations individuelles achetaient leur propre matériel informatique, construisaient leurs propres salles informatiques, acquéraient leurs propres connexions réseau et géraient leur propre logiciel de serveur Web.

Avec la démocratisation rapide du Web, tout le monde voulait un site web, mais peu de gens avaient les compétences ou le temps de construire des salles de serveurs climatisées, d'enregistrer des noms de domaine ou d'acheter de la bande passante réseau. Pour sauver la situation, de nombreuses nouvelles entreprises ont émergé, proposant des services d'hébergement web gérés par des professionnels. Plusieurs niveaux de service sont disponibles, de la gestion des installations physiques (mise à disposition de l'espace, de la climatisation et du câblage) à l'hébergement web complet, où le client se contente de fournir le contenu.

Ce chapitre se concentre sur les fonctionnalités du serveur web d'hébergement. Le bon fonctionnement d'un site web, comme sa prise en charge de différentes langues et la sécurisation des transactions e-commerce, dépend en grande partie des fonctionnalités prises en charge par le serveur web d'hébergement.

Un exemple simple : l'hébergement dédié

Supposons que Joe's Hardware Online et Mary's Antique Auction souhaitent tous deux gérer des sites web à fort volume. Le FAI d'Irène dispose d'une multitude de serveurs web identiques et performants qu'il peut louer à Joe et Mary, au lieu de leur confier l'achat de leurs propres serveurs et la maintenance de leurs logiciels.

Dans la figure 18-1, Joe et Mary souscrivent tous deux à l'hébergement web dédié proposé par le FAI d'Irène. Joe loue un serveur web dédié, acheté et maintenu par le FAI d'Irène. Mary obtient un autre serveur dédié auprès du FAI d'Irène. Le FAI d'Irène peut acheter du matériel serveur en volume et choisir du matériel fiable, éprouvé et économique. Si le site de vente en ligne de matériel de Joe ou la vente aux enchères d'antiquités de Mary gagne en popularité, le FAI d'Irène peut immédiatement proposer des serveurs supplémentaires à Joe ou à Mary.

Figure 18-1. Hébergement dédié externalisé

Dans cet exemple, les navigateurs envoient des requêtes HTTP pour www.joes-hardware.com à l'adresse IP du serveur de Joe et des requêtes pour www.marys-antiques.com à l'adresse IP (différente) du serveur de Mary.

Hébergement virtuel

De nombreuses personnes souhaitent être présentes sur le web, mais n'ont pas de sites web à fort trafic. Pour elles, fournir un serveur web dédié peut être un gaspillage, car elles paient des centaines de dollars par mois pour louer un serveur généralement inactif !

De nombreux hébergeurs web proposent des services d'hébergement web à moindre coût en partageant un ordinateur entre plusieurs clients. On parle alors d'hébergement mutualisé ou d'hébergement virtuel. Chaque site web semble hébergé sur un serveur différent, mais ils sont en réalité hébergés sur le même serveur physique. Du point de vue de l'utilisateur final, les sites web hébergés virtuellement devraient être identiques à ceux hébergés sur des serveurs dédiés distincts.

Pour des raisons de rentabilité, d'espace et de gestion, un hébergeur virtuel souhaite héberger des dizaines, des centaines, voire des milliers de sites web sur un même serveur. Cela ne signifie pas nécessairement que 1 000 sites web sont hébergés sur un seul PC. Les hébergeurs peuvent créer des groupes de serveurs répliqués (appelés fermes de serveurs) et répartir la charge sur l'ensemble de la ferme. Chaque serveur de la ferme étant un clone des autres et hébergeant de nombreux sites web virtuels, l'administration est grandement simplifiée. (Nous reviendrons sur les fermes de serveurs au chapitre 20.)

Lorsque Joe et Mary ont démarré leur entreprise, ils ont peut-être choisi l'hébergement virtuel pour économiser de l'argent jusqu'à ce que leurs niveaux de trafic justifient l'utilisation d'un serveur dédié (voir Figure 18-2).

Figure 18-2. Hébergement virtuel externalisé

La demande de serveur virtuel manque d'informations sur l'hôte

Malheureusement, HTTP/1.0 présente un défaut de conception qui intrigue les hébergeurs virtuels. La spécification HTTP/1.0 ne permettait pas aux serveurs web partagés d'identifier les sites web virtuels qu'ils hébergent et auxquels ils accédaient.

Rappelons que les requêtes HTTP/1.0 n'envoient que le chemin d'accès de l'URL dans le message de requête. Si vous essayez d'obtenir http://www.joes-hardware.com/index.html, le navigateur se connecte au serveur www.joes-hardware.com, mais la requête HTTP/1.0 indique « GET /index.html », sans autre mention du nom d'hôte. Si le serveur héberge virtuellement plusieurs sites, ces informations ne suffisent pas à déterminer à quel site web virtuel l'utilisateur accède. Par exemple, dans la figure 18-3 :

• Si le client A tente d'accéder à http://www.joes-hardware.com/index.html, la requête « GET /index.html » sera envoyée au serveur Web partagé.

• Si le client B tente d'accéder à http://www.marys-antiques.com/index.html, la même requête « GET /index.html » sera envoyée au serveur Web partagé.

Figure 18-3. Les requêtes du serveur HTTP/1.0 ne contiennent pas d'informations sur le nom d'hôte

Concernant le serveur web, les informations sont insuffisantes pour déterminer à quel site web l'utilisateur accède ! Les deux requêtes semblent identiques, même si elles concernent des documents totalement différents (provenant de sites web différents). Le problème est que les informations sur l'hôte du site web ont été supprimées de la requête.

Comme nous l'avons vu au chapitre 6, les substituts HTTP (proxies inverses) et les proxys d'interception ont également besoin d'informations spécifiant le site.

Faire fonctionner l'hébergement virtuel

L'absence d'informations sur l'hôte était due à une erreur dans la spécification HTTP d'origine, qui supposait à tort que chaque serveur web hébergeait un seul site web. Les concepteurs de HTTP n'ont pas pris en charge les serveurs partagés hébergés virtuellement. Pour cette raison, les informations sur le nom d'hôte dans l'URL ont été considérées comme redondantes et supprimées ; seul le chemin d'accès devait être envoyé.

Les premières spécifications ne prévoyant pas l'hébergement virtuel, les hébergeurs web ont dû développer des solutions de contournement et des conventions pour prendre en charge l'hébergement virtuel partagé. Le problème aurait pu être résolu simplement en exigeant que toutes les requêtes HTTP envoient l'URL complète plutôt que le chemin d'accès. HTTP/1.1 exige que les serveurs gèrent les URL complètes dans les lignes de requête des messages HTTP, mais il faudra beaucoup de temps avant que toutes les applications existantes ne soient mises à niveau vers cette spécification. Entre-temps, quatre techniques ont émergé :

Hébergement virtuel par chemin URL

Ajout d'un composant de chemin spécial à l'URL afin que le serveur puisse déterminer le site.

Hébergement virtuel par numéro de port

Attribuer un numéro de port différent à chaque site, afin que les demandes soient traitées par des instances distinctes du serveur Web.

Hébergement virtuel par adresse IP

Affecter des adresses IP différentes à différents sites virtuels et les lier à une seule machine. Cela permet au serveur web d'identifier le nom du site par son adresse IP.

Hébergement virtuel par en-tête d'hôte

De nombreux hébergeurs web ont fait pression sur les concepteurs de HTTP pour résoudre ce problème. Les versions améliorées de HTTP/1.0 et la version officielle de HTTP/1.1 définissent un en-tête de requête Host contenant le nom du site. Le serveur web peut identifier le site virtuel à partir de cet en-tête.

Examinons de plus près chaque technique.

Hébergement virtuel par chemin URL

Vous pouvez utiliser la force brute pour isoler des sites virtuels sur un serveur partagé en leur attribuant des chemins d'URL différents. Par exemple, vous pouvez attribuer à chaque site web logique un préfixe de chemin spécifique :

• La quincaillerie Joe's pourrait être http://www.joes-hardware.com/joe/index.html.

• Le magasin d'antiquités de Mary pourrait être http://www.marys-antiques.com/mary/index.html.

Lorsque les requêtes arrivent au serveur, les informations sur le nom d'hôte ne sont pas présentes dans la requête, mais le serveur peut les distinguer en fonction du chemin :

• La requête pour le matériel de Joe est « GET /joe/index.html ».

• La demande pour les antiquités de Mary est « GET /mary/index.html ».

Ce n'est pas une bonne solution. Les préfixes « /joe » et « /mary » sont redondants et prêtent à confusion (nous avons déjà mentionné « joe » dans le nom d'hôte). Pire encore, la convention courante consistant à spécifier http://www.joes-hardware.com ou http://www.joes-hardware.com/index.html pour la page d'accueil ne fonctionnera pas.

En général, l’hébergement virtuel basé sur une URL est une mauvaise solution et est rarement utilisé.

Hébergement virtuel par numéro de port

Au lieu de modifier le chemin d'accès, Joe et Mary pourraient chacun se voir attribuer un numéro de port différent sur le serveur web. Par exemple, au lieu du port 80, Joe pourrait obtenir le 82 et Mary le 83. Mais cette solution présente le même problème : l'utilisateur final s'attendrait à trouver les ressources sans avoir à spécifier un port non standard dans l'URL.

Hébergement virtuel par adresse IP

Une approche bien plus efficace (et couramment utilisée) est l'adressage IP virtuel. Chaque site web virtuel obtient alors une ou plusieurs adresses IP uniques. Les adresses IP de tous les sites web virtuels sont rattachées au même serveur partagé. Le serveur peut

rechercher l'adresse IP de destination de la connexion HTTP et l'utiliser pour déterminer à quel site web le client pense être connecté.

Imaginons qu'un hébergeur attribue l'adresse IP 209.172.34.3 à www.joes-hardware.com, l'adresse 209.172.34.4 à www.marys-antiques.com, et associe les deux adresses IP au même serveur physique. Le serveur web pourrait alors utiliser l'adresse IP de destination pour identifier le site virtuel demandé, comme illustré à la figure 18-4 :

• Le client A récupère http://www.joes-hardware.com/index.html.

• Le client A trouve l'adresse IP de www.joes-hardware.com et obtient 209.172.34.3.

• Le client A ouvre une connexion TCP au serveur Web partagé à 209.172.34.3.

• Le client A envoie la requête « GET /index.html HTTP/1.0 ».

Avant de fournir une réponse, le serveur Web note l'adresse IP de destination réelle (209.172.34.3), détermine qu'il s'agit d'une adresse IP virtuelle pour le site web de Joe et répond à la requête depuis le sous-répertoire /joe. La page /joe/index.html est renvoyée.

Figure 18-4. Hébergement IP virtuel

De même, si le client B demande http://www.marys-antiques.com/index.html :

• Le client B trouve l'adresse IP de www.marys-antiques.com et obtient 209.172.34.4.

• Le client B ouvre une connexion TCP au serveur Web à l’adresse 209.172.34.4.

• Le client B envoie la requête « GET /index.html HTTP/1.0 ».

• Le serveur Web détermine que 209.172.34.4 est le site Web de Mary et répond à la demande du sous-répertoire /mary, renvoyant le document /mary/index.html.

L'hébergement IP virtuel fonctionne, mais il pose quelques difficultés, notamment pour les grands hébergeurs :

• Les systèmes informatiques ont généralement une limite quant au nombre d'adresses IP virtuelles pouvant être associées à une machine. Les hébergeurs souhaitant héberger des centaines, voire des milliers de sites virtuels sur un serveur mutualisé risquent d'être désavantagés.

• Les adresses IP sont rares. Les hébergeurs disposant de nombreux sites virtuels pourraient ne pas être en mesure d'obtenir suffisamment d'adresses IP virtuelles pour les sites web hébergés.

• La pénurie d'adresses IP est aggravée lorsque les hébergeurs répliquent leurs serveurs pour augmenter leur capacité. Des adresses IP virtuelles différentes peuvent être nécessaires sur chaque serveur répliqué, selon l'architecture d'équilibrage de charge. Le nombre d'adresses IP nécessaires peut donc être multiplié par le nombre de serveurs répliqués.

Malgré les problèmes de consommation d'adresses liés à l'hébergement IP virtuel, celui-ci est largement utilisé.

Hébergement virtuel par en-tête d'hôte

Pour éviter une consommation excessive d'adresses et des limites d'adresses IP virtuelles, nous souhaitons partager la même adresse IP entre les sites virtuels, tout en étant capables de les distinguer. Cependant, comme nous l'avons vu, la plupart des navigateurs n'envoient que le chemin d'accès de l'URL aux serveurs, ce qui entraîne la perte des informations critiques du nom d'hôte virtuel.

Pour résoudre ce problème, les développeurs de navigateurs et de serveurs ont étendu HTTP afin de fournir le nom d'hôte d'origine aux serveurs. Cependant, les navigateurs ne pouvaient pas se contenter d'envoyer une URL complète, car cela aurait endommagé de nombreux serveurs qui ne s'attendaient à recevoir qu'un chemin d'accès. À la place, le nom d'hôte (et le port) sont transmis dans un en-tête d'extension Host dans toutes les requêtes.

Dans la figure 18-5, les clients A et B envoient tous deux des en-têtes Host contenant le nom d'hôte d'origine auquel ils accèdent. Lorsque le serveur reçoit la requête pour /index.html, il peut utiliser l'en-tête Host pour déterminer les ressources à utiliser.

Figure 18-5. Les en-têtes d'hôte distinguent les requêtes d'hôte virtuel

Les en-têtes d'hôte ont été introduits avec HTTP/1.0+, un sur-ensemble étendu de HTTP/1.0 par le fournisseur. Ils sont requis pour la conformité HTTP/1.1. Ils sont pris en charge par la plupart des navigateurs et serveurs modernes, mais certains clients et serveurs (et robots) ne les prennent toujours pas en charge.

En-têtes d'hôte HTTP/1.1

L'en-tête Host est un en-tête de requête HTTP/1.1, défini dans la RFC 2068. Les serveurs virtuels sont si courants que la plupart des clients HTTP, même s'ils ne sont pas compatibles HTTP/1.1, implémentent l'en-tête Host.

Syntaxe et utilisation

L'en-tête Host spécifie l'hôte Internet et le numéro de port de la ressource demandée, tels qu'obtenus à partir de l'URL d'origine :

Hôte = "Hôte" ":" hôte [":" port] En particulier :

• Si l'en-tête de l'hôte ne contient pas de port, le port par défaut du schéma est utilisé.

• Si l'URL contient une adresse IP, l'en-tête Host doit contenir la même adresse.

• Si l'URL contient un nom d'hôte, l'en-tête Host doit contenir le même nom.

• Si l'URL contient un nom d'hôte, l'en-tête Host ne doit pas contenir l'adresse IP équivalente au nom d'hôte de l'URL, car cela endommagerait les serveurs hébergés virtuellement, qui superposent plusieurs sites virtuels sur une seule adresse IP.

• Si l'URL contient un nom d'hôte, l'en-tête Host ne doit pas contenir un autre alias pour ce nom d'hôte, car cela endommagerait également les serveurs hébergés virtuellement.

• Si le client utilise un serveur proxy explicite, il doit inclure le nom et le port du serveur d'origine dans l'en-tête Host, et non celui du serveur proxy. Par le passé, plusieurs clients web présentaient des bugs : l'en-tête Host sortant était défini sur le nom d'hôte du proxy, lorsque le paramètre proxy du client était activé. Ce comportement incorrect entraînait des dysfonctionnements des proxys et des serveurs d'origine.

• Les clients Web doivent inclure un champ d'en-tête Host dans tous les messages de demande.

• Les proxys Web doivent ajouter des en-têtes d'hôte pour demander des messages avant de les transférer.

• Les serveurs Web HTTP/1.1 doivent répondre avec un code d'état 400 à tout message de requête HTTP/1.1 dépourvu de champ d'en-tête Host.

Voici un exemple de message de requête HTTP utilisé pour récupérer la page d'accueil de www.joeshardware.com, ainsi que le champ d'en-tête Host requis :

OBTENIR http://www.joes-hardware.com/index.html HTTP/1.0

Connexion : Keep-Alive

Agent utilisateur : Mozilla/4.51 [fr] (X11; U; IRIX 6.2 IP22)

Accepter : image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*

Accepter-Encodage : gzip

Accepter-Langue : en

Hébergeur : www.joes-hardware.com

En-têtes d'hôte manquants

Un faible pourcentage d'anciens navigateurs n'envoient pas d'en-têtes d'hôte. Si un serveur d'hébergement virtuel utilise des en-têtes d'hôte pour déterminer le site web à héberger, et qu'aucun en-tête d'hôte n'est présent, il dirigera probablement l'utilisateur vers une page web par défaut (comme celle du FAI) ou renverra une page d'erreur lui suggérant de mettre à jour son navigateur.

Interprétation des en-têtes d'hôte

Un serveur d'origine qui n'est pas hébergé virtuellement et qui n'autorise pas la différenciation des ressources selon l'hôte demandé peut ignorer la valeur du champ d'en-tête Hôte. En revanche, tout serveur d'origine qui différencie les ressources selon l'hôte doit utiliser les règles suivantes pour déterminer la ressource demandée lors d'une requête HTTP/1.1 :

1. Si l'URL dans le message de requête HTTP est absolue (c'est-à-dire qu'elle contient un composant de schéma et d'hôte), la valeur dans l'en-tête Host est ignorée en faveur de l'URL.

2. Si l'URL dans le message de requête HTTP n'a pas d'hôte et que la requête contient un en-tête Host, la valeur de l'hôte/port est obtenue à partir de l'en-tête Host.

3. Si aucun hôte valide ne peut être déterminé via les étapes 1 ou 2, une réponse 400 Bad Response est renvoyée au client.

En-têtes d'hôte et proxys

Certaines versions de navigateur envoient des en-têtes d'hôte incorrects, notamment lorsqu'elles sont configurées pour utiliser des proxys. Par exemple, lors de la configuration d'un proxy, certaines anciennes versions de clients Apple et PointCast envoyaient par erreur le nom du proxy au lieu du serveur d'origine dans l'en-tête d'hôte.

Rendre les sites Web fiables

Il existe plusieurs situations dans lesquelles les sites Web tombent en panne :

• Temps d'arrêt du serveur

• Pics de trafic : tout le monde veut soudainement voir un journal télévisé ou se précipiter sur une vente. Des pics soudains peuvent surcharger un serveur web, le ralentir, voire l'arrêter complètement.

• Pannes ou pertes de réseau

Cette section présente quelques moyens d’anticiper et de traiter ces problèmes courants.

Rendre les sites Web fiables

Fermes de serveurs en miroir

Une ferme de serveurs est un ensemble de serveurs web configurés de manière identique, capables de se relayer mutuellement. Le contenu de chaque serveur de la ferme peut être mis en miroir, de sorte qu'en cas de problème sur l'un d'eux, un autre peut prendre le relais.

Les serveurs miroirs suivent souvent une relation hiérarchique. Un serveur peut faire office d'« autorité de contenu » : il héberge le contenu original (par exemple, un serveur sur lequel les auteurs publient). Ce serveur est appelé serveur d'origine maître. Les serveurs miroirs qui reçoivent le contenu du serveur d'origine maître sont appelés serveurs d'origine répliqués. Une méthode simple pour déployer une ferme de serveurs consiste à utiliser un commutateur réseau pour distribuer les requêtes aux serveurs. L'adresse IP de chaque site web hébergé sur les serveurs correspond à l'adresse IP du commutateur.

Dans la batterie de serveurs en miroir illustrée à la figure 18-6, le serveur d'origine maître est responsable de l'envoi du contenu aux serveurs d'origine répliqués. Pour l'extérieur, l'adresse IP de ce contenu est celle du commutateur. Ce dernier est responsable de l'envoi des requêtes aux serveurs.

Figure 18-6. Batterie de serveurs en miroir

Les serveurs web en miroir peuvent contenir des copies du même contenu à différents emplacements. La figure 18-7 illustre quatre serveurs en miroir, avec un serveur maître à Chicago et des serveurs réplicas à New York, Miami et Little Rock. Le serveur maître dessert les clients de la région de Chicago et a également pour mission de propager son contenu vers les serveurs réplicas.

Dans le scénario de la figure 18-7, il existe plusieurs manières de diriger les demandes des clients vers un serveur particulier :

Redirection HTTP

L'URL du contenu pourrait être résolue en l'adresse IP du serveur maître, qui pourrait ensuite envoyer des redirections vers des serveurs réplicas.

Figure 18-7. Serveurs miroirs dispersés

Redirection DNS

L'URL du contenu pourrait être résolue en quatre adresses IP, et le serveur DNS pourrait choisir l'adresse IP qu'il envoie aux clients.

Voir le chapitre 20 pour plus de détails.

Réseaux de distribution de contenu

Un réseau de distribution de contenu (CDN) est un réseau dont l'objectif est la distribution de contenu spécifique. Les nœuds du réseau peuvent être des serveurs web, des serveurs de substitution ou des caches proxy.

Caches de substitution dans les CDN

Les caches de substitution peuvent être utilisés à la place des serveurs d'origine répliqués dans les figures 18-6 et 18-7. Les substituts, également appelés proxys inverses, reçoivent les requêtes de contenu des serveurs, tout comme les serveurs web en miroir. Ils reçoivent les requêtes des serveurs pour le compte d'un ensemble spécifique de serveurs d'origine (cela est possible grâce à la manière dont les adresses IP des contenus sont annoncées ; il existe généralement une relation de travail entre le serveur d'origine et le substitut, et les substituts s'attendent à recevoir des requêtes destinées à des serveurs d'origine spécifiques).

La différence entre un serveur de substitution et un serveur miroir réside dans le fait que les serveurs de substitution sont généralement pilotés par la demande. Ils ne stockent pas de copies complètes du contenu du serveur d'origine ; ils stockent le contenu demandé par les clients. La distribution du contenu dans leurs caches dépend des requêtes reçues ; le serveur d'origine n'est pas responsable de la mise à jour de son contenu. Pour faciliter l'accès au contenu « chaud » (contenu très demandé), certains serveurs de substitution disposent de fonctionnalités de « préchargement » qui leur permettent d'extraire le contenu avant les requêtes des utilisateurs.

Une complexité supplémentaire dans les CDN avec des substituts est la possibilité de hiérarchies de cache.

Rendre les sites Web fiables

Caches proxy dans les CDN

Les caches proxy peuvent également être déployés dans des configurations similaires à celles des figures 18-6 et 18-7. Contrairement aux substituts, les caches proxy traditionnels peuvent recevoir des requêtes destinées à n'importe quel serveur web (aucune relation de travail ni accord d'adresse IP n'est requis entre un cache proxy et un serveur d'origine). Cependant, comme pour les substituts, le contenu du cache proxy est généralement géré à la demande et n'est pas censé être une copie exacte du contenu du serveur d'origine. Certains caches proxy peuvent également être préchargés avec du contenu important.

Les caches proxy pilotés par la demande peuvent être déployés dans d'autres types de configurations, en particulier les configurations d'interception, où un périphérique de couche 2 ou 3 (commutateur ou routeur) intercepte le trafic Web et l'envoie vers un cache proxy (voir Figure 18-8).

Figure 18-8. Requêtes client interceptées par un commutateur et envoyées à un proxy

La configuration d'une interception repose sur la capacité à configurer le réseau entre clients et serveurs afin que toutes les requêtes HTTP appropriées soient physiquement acheminées vers le cache (voir chapitre 20). Le contenu est distribué dans le cache en fonction des requêtes reçues.

Rendre les sites Web rapides

De nombreuses technologies mentionnées dans la section précédente contribuent également à accélérer le chargement des sites web. Les fermes de serveurs et les caches proxy distribués, ou serveurs de substitution, répartissent le trafic réseau, évitant ainsi la congestion. La distribution du contenu le rapproche des utilisateurs finaux, réduisant ainsi le temps de trajet du serveur au client. La rapidité d'accès aux ressources repose sur la manière dont les requêtes et les réponses sont acheminées du client au serveur et inversement via Internet. Consultez le chapitre 20 pour plus de détails sur les méthodes de redirection.

Une autre approche pour accélérer les sites web consiste à encoder le contenu pour un transport rapide. Cela peut impliquer, par exemple, de le compresser, en supposant que le client destinataire puisse le décompresser. Voir le chapitre 15 pour plus de détails.

Pour plus d'informations

Consultez la partie III, Identification, autorisation et sécurité, pour plus de détails sur la sécurisation des sites web. Les documents et ébauches de sites web suivants peuvent vous fournir plus de détails sur l'hébergement web et la distribution de contenu :

http://www.ietf.org/rfc/rfc3040.txt

La RFC 3040, « Internet Web Replication and Caching Taxonomy », est une référence pour le vocabulaire des applications de réplication et de mise en cache Web.

http://search.ietf.org/internet-drafts/draft-ietf-cdi-request-routing-reqs-00.txt « Exigences de routage des requêtes pour l'interconnexion de contenu. »

Ben Laurie et Peter Laurie, O'Reilly & Associates, Inc. Ce livre décrit comment exécuter le serveur Web open source Apache.

Pour plus d'informations

Chapitre 19Ceci est le titre du livreCHAPITRE 19

Systèmes d'édition

Comment créer des pages web et les transférer sur un serveur web ? Aux temps anciens du Web (disons en 1995), vous auriez pu rédiger votre code HTML à la main dans un éditeur de texte et télécharger manuellement le contenu sur le serveur web via FTP. Cette procédure était fastidieuse, difficile à coordonner avec vos collègues et peu sécurisée.

Les outils de publication modernes simplifient grandement la création, la publication et la gestion de contenu web. Aujourd'hui, vous pouvez modifier interactivement le contenu web tel qu'il apparaît à l'écran et le publier sur des serveurs d'un simple clic, tout en étant informé des modifications apportées aux fichiers.

De nombreux outils prenant en charge la publication de contenu à distance utilisent des extensions du protocole HTTP. Dans ce chapitre, nous expliquons deux technologies importantes pour la publication de contenu web basée sur HTTP : FrontPage et DAV.

Prise en charge des extensions serveur FrontPage pour la publication

FrontPage (communément appelé FP) est une boîte à outils polyvalente de création et de publication Web fournie par Microsoft Corp. L'idée originale de FrontPage (FrontPage 1.0) a été conçue en 1994 par Vermeer Technologies, Inc. et a été considérée comme le premier produit à combiner la gestion et la création de sites Web en un seul outil unifié. Microsoft a racheté Vermeer et a commercialisé FrontPage 1.1 en 1996. La dernière version, FrontPage Version 2002, est la sixième de la gamme et un élément essentiel de la suite.

Suite Microsoft Office.

Extensions serveur FrontPage

Dans le cadre de sa stratégie de publication en tout lieu, Microsoft a lancé un ensemble de logiciels côté serveur appelés FrontPage Server Extensions (FPSE). Ces composants côté serveur s'intègrent au serveur web et assurent la traduction nécessaire entre le site web et le client exécutant FrontPage (et les autres clients prenant en charge ces extensions).

Notre principal intérêt réside dans le protocole de publication entre les clients FP et FPSE. Ce protocole fournit un exemple de conception d'extensions aux services principaux disponibles en HTTP sans modifier la sémantique HTTP.

Le protocole de publication FrontPage implémente une couche RPC au-dessus de la requête HTTP POST. Cela permet au client FrontPage d'envoyer des commandes au serveur pour mettre à jour les documents du site web, effectuer des recherches, collaborer entre les auteurs web, etc. La figure 19-1 donne un aperçu de la communication.

Figure 19-1. Architecture de publication FrontPage

Le serveur Web détecte les requêtes POST adressées au FPSE (implémenté sous forme d'un ensemble de programmes CGI, dans le cas d'un serveur IIS autre que Microsoft) et les oriente en conséquence. Tant que les pare-feu et les serveurs proxy sont configurés pour autoriser la méthode POST, FrontPage peut continuer à communiquer avec le serveur.

Vocabulaire FrontPage

Avant de plonger plus en profondeur dans la couche RPC définie par FPSE, il peut être utile d'établir le vocabulaire commun :

Serveur virtuel

Un site web parmi tant d'autres fonctionnant sur le même serveur, chacun possédant un nom de domaine et une adresse IP uniques. En substance, un serveur virtuel permet à un seul serveur web d'héberger plusieurs sites web, chacun apparaissant au navigateur comme hébergé par son propre serveur web. Un serveur web prenant en charge les serveurs virtuels est appelé serveur web multi-hébergement. Une machine configurée avec plusieurs adresses IP est appelée serveur multi-hébergé (pour plus de détails, consultez la section « Hébergement virtuel » au chapitre 18).

Toile racinaire

Répertoire de contenu principal par défaut d'un serveur web ou, dans un environnement multi-hébergement, répertoire de contenu principal d'un serveur web virtuel. Pour accéder au répertoire racine, il suffit de spécifier l'URL du serveur sans spécifier de nom de page. Il ne peut y avoir qu'un seul répertoire racine par serveur web.

Prise en charge des extensions serveur FrontPage pour la publication

Sous-site Web

Un sous-répertoire nommé du site web racine ou d'un autre sous-site web constituant un site web FPSE étendu complet. Un sous-site web peut être une entité totalement indépendante, capable de définir ses

propres autorisations d'administration et de création. De plus, les sous-site web peuvent définir la portée de méthodes telles que les recherches.

Le protocole RPC de FrontPage

Le client FrontPage et FPSE communiquent via un protocole RPC propriétaire. Ce protocole se superpose à HTTP POST en intégrant les méthodes RPC et leurs variables associées dans le corps de la requête POST.

Pour démarrer le processus, le client doit déterminer l'emplacement et le nom des programmes cibles sur le serveur (la partie du package FPSE qui peut exécuter le

(requête POST). Il émet ensuite une requête GET spéciale (voir Figure 19-2).

Figure 19-2. Demande initiale

Lorsque le fichier est renvoyé, le client FrontPage lit la réponse et trouve les valeurs associées à FPShtmlScriptUrl, FPAuthorScriptUrl et FPAdminScriptUrl. Voici un exemple typique :

FPShtmlScriptUrl="_vti_bin/_vti_rpc/shtml.dll"

FPAuthorScriptUrl="_vti_bin/_vti_aut/author.dll"

FPAdminScriptUrl="_vti_bin/_vti_adm/admin.dll"

FPShtmlScriptUrl indique au client où POSTER les requêtes pour les commandes « de navigation » (par exemple, obtenir la version de FPSE) à exécuter.

FPAuthorScriptUrl indique au client où envoyer les requêtes POST pour les commandes de création à exécuter. De même, FPAdminScriptUrl indique à FrontPage où envoyer les requêtes pour les actions administratives.

Maintenant que nous savons où se trouvent les différents programmes, nous sommes prêts à envoyer une demande.

Demande

Le corps de la requête POST contient la commande RPC, sous la forme « method=<command> », et tous les paramètres requis. Prenons l'exemple de

Message RPC demandant une liste de documents, comme suit :

POST /_vti_bin/_vti_aut/author.dll HTTP/1.1

Date : sam. 12 août 2000 20:32:54 GMT

Agent utilisateur : MSFrontPage/4.0 .....

<CORPS>

méthode=liste+documents%3a4%2e0%2e2%2e3717&service%5fname
=&listHiddenDocs=false&listExplorerDocs=false&listRecurse=false&listF
iles=true&listFolders=true&listLinkInfo=true&listIncludeParent=true&li
stDerived=false

&listBorders=false&listChildWebs=true&initialUrl=&folderList=%5b%3b
TW%7c12+août+2000+2

0%3a33%3a04+%2d0000%5d

Le corps de la commande POST contient la commande RPC envoyée au FPSE. Comme pour les programmes CGI, les espaces de la méthode sont codés avec le signe plus (+). Tous les autres caractères non alphanumériques de la méthode sont codés au format %XX, où XX représente la représentation ASCII du caractère. Avec cette notation, une version plus lisible du corps ressemblerait à ceci :

méthode=liste+documents:4.0.1.3717

&service_name=

&listHiddenDocs=false

&listExplorerDocs=false .....

Certains des éléments énumérés sont :

nom_service

L'URL du site web sur lequel la méthode doit agir. Elle doit correspondre à un dossier existant ou à un niveau inférieur.

listeHiddenDocs

Affiche les documents masqués d'un site web si sa valeur est « true ». Les documents masqués sont désignés par des URL dont les chemins commencent par « _ ».

listeExploreDocs

Si la valeur est « vrai », répertorie les listes de tâches.

Réponse

La plupart des méthodes du protocole RPC ont des valeurs de retour. Les valeurs de retour les plus courantes concernent les méthodes réussies et les erreurs. Certaines méthodes comportent également une troisième sous-section, « Exemple de code de retour ». FrontPage interprète correctement les codes pour fournir un retour précis à l'utilisateur.

Prise en charge des extensions serveur FrontPage pour la publication

Poursuivons avec notre exemple : le FPSE traite la requête « liste+documents » et renvoie les informations nécessaires. Voici un exemple de réponse :

HTTP/1.1 200 OK

Serveur : Microsoft-IIS/5.0

Date : sam. 12 août 2000 22:49:50 GMT

Type de contenu : application/x-vermeer-rpc

Nom d'utilisateur X-FrontPage : IUSER_MINSTAR

<html><head><title>Paquet RPC</title></head>

<corps>

<p>méthode=liste des documents : 4.0.2.3717

<p>document_list=

document_name=help.gif

<\ul>

Comme vous pouvez le constater dans la réponse, une liste formatée des documents disponibles sur le serveur web est renvoyée au client FP. Vous trouverez la liste complète des commandes et des réponses sur le site web de Microsoft.

Modèle de sécurité FrontPage

Tout système de publication accédant directement au contenu d'un serveur web doit être parfaitement conscient des implications de ses actions en matière de sécurité. FPSE dépend principalement du serveur web pour assurer la sécurité.

Le modèle de sécurité FPSE définit trois types d'utilisateurs : administrateurs, auteurs et navigateurs. Les administrateurs disposent d'un contrôle total. Toutes les autorisations sont cumulatives ; tous les administrateurs peuvent créer et naviguer sur le site web FrontPage. De même, tous les auteurs disposent d'autorisations de navigation.

La liste des administrateurs, auteurs et navigateurs est définie pour un site web étendu FPSE donné. Tous les sous-sites peuvent hériter des autorisations du site web racine ou définir leurs propres autorisations. Pour les serveurs web non IIS, tous les programmes FPSE doivent être stockés dans des répertoires marqués « exécutables » (même restriction que pour tout autre programme CGI). Fpsrvadm, l'utilitaire d'administration du serveur FrontPage, peut être utilisé à cette fin. Sur les serveurs IIS, le modèle de sécurité intégré de Windows prévaut.

Sur les serveurs non IIS, les mécanismes de contrôle d'accès des serveurs web spécifient les utilisateurs autorisés à accéder à un programme donné. Sur les serveurs web Apache et NCSA, le fichier s'appelle .htaccess ; sur les serveurs Netscape, il s'appelle .nsconfig. Ce fichier d'accès associe les utilisateurs, les groupes et les adresses IP à différents niveaux d'autorisation : GET (lecture), POST (exécution), etc. Par exemple, pour qu'un utilisateur soit auteur sur un serveur web Apache, le fichier .htaccess doit lui permettre d'envoyer des messages POST vers author.exe. Ces fichiers de spécification d'accès sont souvent définis répertoire par répertoire, offrant une plus grande flexibilité dans la définition des autorisations.

Sur les serveurs IIS, les autorisations sont vérifiées par rapport aux listes de contrôle d'accès (ACL) d'une racine ou d'une sous-racine donnée. Lorsqu'IIS reçoit une requête, il se connecte d'abord et emprunte l'identité de l'utilisateur, puis envoie la requête à l'une des trois bibliothèques de liens dynamiques (DLL) d'extension. La DLL compare les informations d'identification d'emprunt à l'ACL définie pour le dossier de destination. Si la vérification est réussie, l'opération demandée est exécutée par la DLL d'extension. Sinon, un message « autorisation refusée » est renvoyé au client. Compte tenu de l'intégration étroite de la sécurité Windows avec IIS, le Gestionnaire d'utilisateurs peut être utilisé pour définir un contrôle précis.

Malgré ce modèle de sécurité élaboré, l'activation de FPSE est devenue un risque de sécurité non négligeable. Dans la plupart des cas, cela est dû à des pratiques négligentes adoptées par les administrateurs de sites web. Cependant, les versions antérieures de FPSE présentaient de graves failles de sécurité, contribuant ainsi à la perception générale d'un risque de sécurité. Ce problème était également exacerbé par les pratiques obscures nécessaires à la mise en œuvre complète d'un modèle de sécurité rigoureux.

WebDAV et création collaborative

WebDAV (Web Distributed Authoring and Versioning) ajoute une dimension supplémentaire à la publication web : la collaboration. Actuellement, la pratique collaborative la plus courante est résolument rudimentaire : principalement par courrier électronique, parfois associé à des partages de fichiers distribués. Cette pratique s'est avérée très peu pratique et sujette aux erreurs, avec peu ou pas de contrôle sur le processus. Prenons l'exemple du lancement d'un site web multinational et multilingue pour un constructeur automobile. On comprend aisément la nécessité d'un système robuste doté de primitives de publication sécurisées et fiables, ainsi que de primitives de collaboration telles que le verrouillage et la gestion des versions.

WebDAV (publié sous la référence RFC 2518) vise à étendre HTTP afin de fournir une plateforme adaptée à la création collaborative. Il est

actuellement développé par l'IETF et bénéficie du soutien de divers fournisseurs, dont Adobe, Apple, IBM, Microsoft, Netscape, Novell, Oracle et Xerox.

Méthodes WebDAV

WebDAV définit un ensemble de nouvelles méthodes HTTP et modifie le champ d'application de plusieurs autres méthodes HTTP. Les nouvelles méthodes ajoutées par WebDAV sont :

PROPFIND

Récupère les propriétés d'une ressource.

PROPPATCH

Définit une ou plusieurs propriétés sur une ou plusieurs ressources.

MKCOL

Crée des collections.

COPIE

Copie une ressource ou un ensemble de ressources d'une source vers une destination donnée. La destination ne doit pas nécessairement se trouver sur la même machine.

SE DÉPLACER

Déplace une ressource ou un ensemble de ressources d'une source vers une destination donnée. La destination ne doit pas nécessairement se trouver sur la même machine.

VERROUILLAGE

Verrouille une ressource ou plusieurs ressources.

OUVRIR

Déverrouille une ressource précédemment verrouillée.

Les méthodes HTTP modifiées par WebDAV sont DELETE, PUT et OPTIONS. Ces méthodes, nouvelles et modifiées, sont détaillées plus loin dans ce chapitre.

WebDAV et XML

Les méthodes WebDAV nécessitent généralement un volume important d'informations pour associer les requêtes et les réponses. HTTP communique généralement ces informations dans les en-têtes de message. Cependant, le transport des informations nécessaires dans les seuls en-têtes impose certaines limitations, notamment la difficulté d'appliquer sélectivement les informations d'en-tête à plusieurs ressources d'une requête, pour représenter une hiérarchie, etc.

Pour résoudre ce problème, WebDAV utilise le langage XML (Extensible Markup Language), un méta-langage de balisage qui fournit un format pour décrire les données structurées. XML offre à WebDAV :

• Une méthode de formatage des instructions décrivant comment les données doivent être traitées

• Une méthode de formatage des réponses complexes du serveur

• Une méthode de communication d'informations personnalisées sur les collections et les ressources gérées

• Un véhicule flexible pour les données elles-mêmes

• Une solution robuste pour la plupart des problèmes d'internationalisation

Traditionnellement, la définition de schéma des documents XML est conservée dans un fichier de définition de type de document (DTD) référencé dans le document XML lui-même. Par conséquent, lors de l'interprétation d'un document XML, l'entité de définition DOCTYPE indique le nom du fichier DTD associé au document XML concerné.

WebDAV définit un espace de noms XML explicite, « DAV: ». Sans entrer dans les détails, un espace de noms XML est un ensemble de noms d'éléments ou d'attributs. Cet espace de noms qualifie les noms intégrés de manière unique dans le domaine, évitant ainsi toute collision de noms.

Le schéma XML complet est défini dans la spécification WebDAV, RFC 2518. La présence d'un schéma prédéfini permet au logiciel d'analyse de faire des hypothèses sur le schéma XML sans avoir à lire les fichiers DTD et à les interpréter correctement.

En-têtes WebDAV

WebDAV introduit plusieurs en-têtes HTTP pour optimiser les fonctionnalités des nouvelles méthodes. Cette section en fournit un bref aperçu ; voir la RFC 2518 pour plus d'informations. Les nouveaux en-têtes sont :

DAV

Utilisé pour communiquer les fonctionnalités WebDAV du serveur. Toutes les ressources prises en charge par WebDAV sont requises pour renvoyer cet en-tête dans la réponse à la requête OPTIONS. Voir « La méthode OPTIONS » pour plus de détails.

DAV = "DAV" ":" "1" ["," "2"] ["," 1#extend]

Profondeur

L'élément crucial pour étendre WebDAV aux ressources groupées avec plusieurs niveaux de hiérarchie (pour une explication plus détaillée sur

les collections, veuillez vous référer à « Gestion des collections et des espaces de noms »).

Profondeur = "Profondeur" ":" ("0" | "1" | "infini")

Prenons un exemple simple. Prenons un répertoire DIR_A contenant les fichiers file_1.html et file_2.html. Si une méthode utilise Depth: 0, elle s'applique uniquement au répertoire DIR_A, tandis que Depth: 1 s'applique au répertoire DIR_A et à ses fichiers file_1.html et file_2.html.

L'en-tête Depth modifie de nombreuses méthodes WebDAV. Parmi les méthodes qui l'utilisent, on trouve LOCK, COPY et MOVE.

Destination

Défini pour aider les méthodes COPY ou MOVE à identifier l'URI de destination.

Destination = "Destination" ":" absoluteURI

Si

Le seul jeton d'état défini est un jeton de verrouillage (voir « La méthode LOCK »). L'en-tête If définit un ensemble de conditions ; si elles sont toutes fausses, la requête échoue. Des méthodes telles que COPY et PUT conditionnent l'applicabilité en spécifiant des conditions préalables dans l'en-tête If. En pratique, la condition préalable la plus courante à satisfaire est l'acquisition préalable d'un verrou.

Si = "Si" ":" (1*Liste sans balise | 1*Liste balisée)

No-tag-list = Liste

Liste étiquetée = Ressource 1*Liste

Ressource = URL codée

Liste = "(" 1*(["Non"](Jeton d'état | "[" balise d'entité "]")) ")"

Jeton d'état = URL codée

URL codée = "<" absoluteURI ">"

Jeton de verrouillage

Utilisé par la méthode UNLOCK pour spécifier le verrou à lever. Une réponse à une méthode LOCK contient également un en-tête Lock-Token, contenant les informations nécessaires sur le verrou levé.

Lock-Token = "Lock-Token" ":" URL codée

Écraser

Utilisé par les méthodes COPY et MOVE pour indiquer si la destination doit être écrasée. Pour plus de détails, voir la discussion sur les méthodes COPY et MOVE plus loin dans ce chapitre.

Écraser = "Écraser" ":" ("V" | "F")

Temps mort

En-tête de requête utilisé par un client pour spécifier la valeur de délai d'expiration de verrouillage souhaitée. Pour plus d'informations, consultez la section « Actualisations de verrouillage et en-tête de délai d'expiration ».

TimeOut = "Timeout" ":" 1#TimeType

TimeType = ("Seconde-" DAVTimeOutVal | "Infini" | Autre)

DAVTimeOutVal = 1*chiffre

Autre = valeur du champ « Étendre »

Maintenant que nous avons esquissé l'intention et la mise en œuvre de WebDAV, examinons de plus près les fonctions fournies.

Verrouillage et prévention de l'écrasement WebDAV

Par définition, la collaboration nécessite la collaboration de plusieurs personnes sur un document donné. Le problème inhérent à la collaboration est illustré à la figure 19-3.

Figure 19-3. Problème de perte de mise à jour

Dans cet exemple, les auteurs A et B rédigent conjointement une spécification. A et B apportent indépendamment une série de modifications au document. A envoie le document mis à jour au référentiel, puis B y publie sa propre version. Malheureusement, B n'ayant jamais eu connaissance des modifications de A, elle n'a jamais fusionné sa version avec celle de A, ce qui a entraîné la perte du travail de ce dernier.

Pour résoudre ce problème, WebDAV prend en charge le concept de verrouillage. Le verrouillage seul ne résoudra pas totalement le problème. La gestion des versions et la prise en charge de la messagerie sont nécessaires pour compléter la solution.

WebDAV prend en charge deux types de verrous :

• Verrouillage exclusif en écriture d'une ressource ou d'une collection

• Verrouillage en écriture partagée d'une ressource ou d'une collection

Un verrou d'écriture exclusif garantit les droits d'écriture uniquement à son propriétaire. Ce type de verrouillage élimine totalement les conflits potentiels. Un verrou d'écriture partagé permet à un groupe de personnes de travailler sur un document donné. Ce type de verrouillage fonctionne bien dans un environnement où tous les auteurs sont informés des activités des autres. WebDAV fournit un mécanisme de découverte de propriétés, via PROPFIND, pour

déterminer la prise en charge du verrouillage et les types de verrous pris en charge.

WebDAV dispose de deux nouvelles méthodes pour prendre en charge le verrouillage : LOCK et UNLOCK.

Pour réaliser le verrouillage, un mécanisme d'identification de l'auteur est nécessaire. WebDAV requiert une authentification Digest (voir chapitre 13).

Lorsqu'un verrou est accordé, le serveur renvoie au client un jeton unique sur le domaine. La spécification désigne ce schéma par le terme « opaquelocktoken ». Lorsque le client souhaite ensuite effectuer une écriture, il se connecte au serveur et complète la séquence d'authentification Digest. Une fois l'authentification terminée, le client WebDAV présente le jeton de verrouillage, accompagné de la requête PUT. Ainsi, la combinaison de l'utilisateur et du jeton de verrouillage est requise pour terminer l'écriture.

La méthode LOCK

Une fonctionnalité puissante de WebDAV est sa capacité à verrouiller plusieurs ressources avec une seule requête LOCK. Le verrouillage WebDAV ne nécessite pas que le client reste connecté au serveur.

Par exemple, voici une simple requête LOCK :

VERROUILLER /ch-publish.fm HTTP/1.1

Hôte : minstar

Type de contenu : texte/xml

Agent utilisateur : Mozilla/4.0 (compatible ; MSIE 5.0 ; Windows NT)

Contenu-Longueur : 201

<?xml version="1.0"?>

<a:lockinfo xmlns:a="DAV:">

<a:lockscope><a:exclusive/></a:lockscope>

<a:locktype><a:write/></a:locktype>

<a:owner><a:href>AuteurA</a:href></a:owner>

</a:lockinfo>

Le XML soumis est basé sur l'élément <lockinfo>. Cette structure comprend trois sous-éléments :

<type de verrouillage>

Indique le type de verrouillage. Il n'en existe actuellement qu'un : « écriture ».

<lunette de verrouillage>

Indique s'il s'agit d'un verrou exclusif ou d'un verrou partagé.

<propriétaire>

Le champ est défini avec la personne qui détient le verrou actuel.

Voici une réponse réussie à notre demande LOCK :

HTTP/1.1 200 OK

Serveur : Microsoft-IIS/5.0

Date : ven. 10 mai 2002 20:56:18 GMT

Type de contenu : texte/xml

Contenu-Longueur : 419

<?xml version="1.0"?>

<a:prop xmlns:a="DAV:">

<a:lockdiscovery><a:activelock>

<a:locktype><a:write/></a:locktype>

<a:lockscope><a:exclusive/></a:lockscope>

<a:owner xmlns:a="DAV:"><a:href>AutherA</a:href></a:owner>

<a:locktoken><a:href>opaquelocktoken:*****</a:href></a:locktoken>

<a:depth>0</a:depth>

<a:timeout>Seconde-180</a:timeout>

</a:activelock></a:lockdiscovery>

</a:prop>

L'élément <lockdiscovery> sert de conteneur pour les informations relatives au verrou. Il intègre un sous-élément <activelock> qui contient les informations envoyées avec la requête (<locktype>, <lockscope> et <owner>). De plus, <activelock> comporte les sous-éléments suivants :

<jeton de verrouillage>

Identifie de manière unique le verrou dans un schéma URI appelé opaquelocktoken. Étant donné la nature sans état du protocole HTTP, ce jeton permet d'identifier le propriétaire du verrou lors des requêtes ultérieures.

<profondeur>

Reflète la valeur de l'en-tête Depth.

<délai d'expiration>

Indique le délai d'expiration associé au verrou. Dans la réponse ci-dessus

(Figure 19-3), la valeur du délai d'attente est de 180 secondes.

Le schéma opaquelocktoken

Le système opaquelocktoken est conçu pour fournir un jeton unique pour toutes les ressources et à tout moment. Pour garantir l'unicité, la spécification WebDAV impose l'utilisation du mécanisme d'identifiant unique universel (UUID), tel que décrit dans la norme ISO-11578.

Concernant l'implémentation concrète, une certaine marge de manœuvre est disponible. Le serveur peut choisir de générer un UUID pour chaque requête LOCK, ou d'en générer un seul et de préserver son unicité en ajoutant des caractères supplémentaires à la fin. Pour des raisons de performances, cette dernière option est préférable. Cependant, si le serveur choisit cette dernière option, il doit garantir qu'aucune des extensions ajoutées ne sera réutilisée.

L'élément XML <lockdiscovery>

L'élément XML <lockdiscovery> fournit un mécanisme de détection active des verrous. Si d'autres personnes tentent de verrouiller le fichier alors qu'un verrou est en place, elles recevront un élément XML <lockdiscovery> indiquant le propriétaire actuel. Cet élément répertorie tous les verrous en attente, ainsi que leurs propriétés.

Verrouiller les actualisations et l'en-tête Timeout

Pour actualiser un verrou, le client doit soumettre à nouveau une demande de verrouillage avec le jeton de verrouillage dans l'en-tête « If ». La valeur du délai d'expiration renvoyée peut être différente des valeurs précédentes.

Au lieu d'accepter la valeur de délai d'expiration fournie par le serveur, le client peut indiquer la valeur de délai d'expiration requise dans la requête LOCK. Cela se fait via l'en-tête Timeout. La syntaxe de l'en-tête Timeout permet au client de spécifier quelques options dans une liste séparée par des virgules. Par exemple :

Délai d'expiration : infini, seconde-86400

Le serveur n'est pas tenu d'accepter ces deux options. Cependant, il est tenu de fournir le délai d'expiration du verrou dans l'élément XML <timeout>. Dans tous les cas, le délai d'expiration du verrou n'est qu'une indication et n'est pas nécessairement contraignant pour le serveur. L'administrateur peut effectuer une réinitialisation manuelle, ou un autre événement exceptionnel peut entraîner la réinitialisation du verrou par le serveur. Les clients doivent éviter les verrous prolongés.

Malgré ces primitives, nous ne parviendrons peut-être pas à résoudre complètement le problème de perte de mise à jour illustré à la figure 19-3. Pour le résoudre complètement, un système d'événements coopératif avec contrôle de version est nécessaire.

La méthode UNLOCK

La méthode UNLOCK supprime un verrou sur une ressource, comme suit :

DÉVERROUILLER /ch-publish.fm HTTP/1.1

Hébergeur : minstar.inktomi.com

Agent utilisateur : Mozilla/4.0 (compatible ; MSIE 5.0 ; Windows NT)

Jeton de verrouillage : opaquelocktoken:*********

HTTP/1.1 204 OK

Serveur : Microsoft-IIS/5.0

Date : ven. 10 mai 2002 20:56:18 GMT

Comme pour la plupart des demandes de gestion des ressources, WebDAV a deux exigences pour

DÉVERROUILLER pour réussir : achèvement préalable d'une séquence d'authentification digest réussie et correspondance avec le jeton de verrouillage envoyé dans l'en-tête Lock-Token.

Si le déverrouillage réussit, un code d'état 204 « Aucun contenu » est renvoyé au client. Le tableau 19-1 récapitule les codes d'état possibles avec les méthodes LOCK et UNLOCK.

Tableau 19-1. Codes d'état pour les méthodes LOCK et UNLOCK

Code d'état défini par la méthode Effet

200 OK HTTP LOCK Indique un verrouillage réussi.

201 Verrou HTTP créé Indique qu'un verrou sur une ressource inexistante a réussi en créant la ressource.

204 Aucun contenu HTTP UNLOCK Indique un déverrouillage réussi.

207 Verrouillage WebDAV multi-états. La requête visait à verrouiller plusieurs ressources. Les codes d'état renvoyés n'étaient pas tous identiques. Ils sont donc tous encapsulés dans une réponse 207.

403 Verrouillage HTTP interdit Indique que le client n'a pas l'autorisation de verrouiller la ressource.

412 Échec de la précondition HTTP LOCK Soit le XML envoyé avec la commande LOCK indiquait une condition à satisfaire et le serveur n'a pas réussi à remplir la condition requise, soit le jeton de verrouillage n'a pas pu être appliqué.

422 Propriété non traitable WebDAV LOCK Sémantique inapplicable : un exemple peut être la spécification d'une profondeur différente de zéro pour une ressource qui n'est pas une collection.

423 WebDAV verrouillé LOCK Déjà verrouillé.

424 Dépendance défaillante. WebDAV UNLOCK spécifie d'autres actions et leur réussite comme condition de déverrouillage. Cette erreur est renvoyée si la dépendance échoue.

Propriétés et métadonnées

Les propriétés décrivent des informations sur la ressource, notamment le nom de l'auteur, la date de modification, la classification du contenu, etc. Les balises META en HTML fournissent un mécanisme permettant d'intégrer ces informations dans le cadre du contenu ; cependant, de nombreuses ressources (telles que les données binaires) n'ont pas la capacité d'intégrer des données META.

Un système collaboratif distribué tel que WebDAV complexifie les exigences en matière de propriétés. Prenons l'exemple d'une propriété d'auteur : lorsqu'un document est modifié, cette propriété doit être mise à jour pour refléter les nouveaux auteurs. WebDAV qualifie ces propriétés modifiables dynamiquement de « propriétés actives ». Les propriétés statiques, plus permanentes, comme Content-Type, sont qualifiées de « propriétés mortes ».

Pour prendre en charge la découverte et la modification des propriétés, WebDAV étend HTTP avec deux nouvelles méthodes : PROPFIND et PROPPATCH. Des exemples et les éléments XML correspondants sont décrits dans les sections suivantes.

La méthode PROPFIND

La méthode PROPFIND (recherche de propriétés) permet de récupérer les propriétés d'un fichier donné ou d'un groupe de fichiers (également appelé « collection »). PROPFIND prend en charge trois types d'opérations :

• Demandez toutes les propriétés et leurs valeurs.

• Demander un ensemble sélectionné de propriétés et de valeurs.

• Demander tous les noms de propriétés.

Voici le scénario dans lequel toutes les propriétés et leurs valeurs sont demandées :

PROPFIND /ch-publish.fm HTTP/1.1

Hébergeur : minstar.inktomi.com

Agent utilisateur : Mozilla/4.0 (compatible ; MSIE 5.0 ; Windows NT)

Profondeur : 0

Contrôle du cache : pas de cache

Connexion : Keep-Alive

Contenu-Longueur : 0

L'élément de requête <propfind> spécifie les propriétés à renvoyer par une méthode PROPFIND. La liste suivante résume quelques éléments XML utilisés avec les requêtes PROPFIND :

<allprop>

Exige que tous les noms et valeurs de propriétés soient renvoyés. Pour demander toutes les propriétés et leurs valeurs, un client WebDAV peut soit envoyer un sous-élément XML <allprop> dans l'élément <propfind>, soit soumettre une requête sans corps.

<propname>

Spécifie l'ensemble des noms de propriétés à renvoyer.

<prop>

Un sous-élément de l'élément <propfind>. Spécifie une propriété spécifique dont la valeur doit être renvoyée. Par exemple : « <a:prop> <a:owner />..... </a:prop> ».

Voici une réponse à un exemple de requête PROPFIND :

HTTP/1.1 207 Multi-Statut

Serveur : Microsoft-IIS/5.0 .....

<?xml version="1.0"?>

<a:multistatusxmlns:b="urn:uuid:********/" xmlns:c="xml:" xmlns:a="DAV:">

<a:response>

<a:href>http://minstar/ch-publish.fm </a:href>

<a:propstat>

<a:status>HTTP/1.1 200OK</a:status>

<a:prop>

<a:getcontentlength b:dt="int">1155</a:getcontentlength>
.....

.....

<a:ishidden b:dt="boolean">0</a:ishidden>

<a:iscollection b:dt="boolean">0</a:iscollection>

</a:prop>

</a:propstat>

</a:response></a:multistatus>

Dans cet exemple, le serveur répond avec un code multi-état 207. WebDAV utilise la réponse 207 pour PROPFIND et quelques autres méthodes WebDAV qui agissent simultanément sur plusieurs ressources et ont potentiellement des réponses différentes pour chaque ressource.

Quelques éléments XML dans la réponse doivent être définis :

<multistatus>

Un conteneur pour plusieurs réponses.

<href>

Identifie l'URI de la ressource.

<statut>

Contient le code d'état HTTP pour la demande particulière.

<propstat>

Regroupe un élément <status> et un élément <prop>. L'élément <prop> peut contenir une ou plusieurs paires nom/valeur de propriété pour la ressource donnée.

Dans l'exemple de réponse ci-dessus, la réponse concerne un URI : http://minstar/chpublish.fm. L'élément <propstat> intègre un élément <status> et un élément <prop>. Pour cet URI, le serveur a renvoyé une réponse 200 OK, comme défini par l'élément <status>. L'élément <prop> comporte plusieurs sous-éléments ; seuls quelques-uns sont répertoriés dans l'exemple.

Une application immédiate de PROPFIND est la prise en charge du listage des répertoires. Grâce à la possibilité d'expression d'une requête PROPFIND, un seul appel permet de récupérer l'intégralité de la hiérarchie de la collection, avec toutes les propriétés des entités individuelles.

La méthode PROPPATCH

La méthode PROPPATCH fournit un mécanisme atomique permettant de définir ou de supprimer plusieurs propriétés d'une ressource donnée. L'atomicité garantit que toutes les requêtes aboutissent ou qu'aucune n'aboutit.

L'élément XML de base de la méthode PROPPATCH est <propertyupdate>. Il sert de conteneur pour toutes les propriétés à mettre à jour. Les éléments XML <set> et <remove> permettent de spécifier l'opération :

<ensemble>

Spécifie les valeurs des propriétés à définir. L'élément <set> contient un ou plusieurs sous-éléments <prop>, qui contiennent à leur tour les paires nom/valeur des propriétés à définir pour la ressource. Si la propriété existe déjà, la valeur est remplacée. <remove>

Spécifie les propriétés à supprimer. Contrairement à <set>, seuls les noms des propriétés sont répertoriés dans le conteneur <prop>.

Cet exemple trivial définit et supprime la propriété « propriétaire » :

```
<d:propertyupdate xmlns:d="DAV:" xmlns:o="http://name-space/scheme/">
 <d:set>
  <d:prop>
   <o:owner>Auteur A</o:owner>
  </d:prop>
 </d:set>
 <d:remove>
  <d:prop>
   <o:propriétaire/>
  </d:prop>
 </d:remove>
</d:propertyupdate>
```

La réponse aux requêtes PROPPATCH est très similaire à celle des requêtes PROPFIND. Pour plus d'informations, consultez la RFC 2518.

Le tableau 19-2 résume les codes d’état des méthodes PROPFIND et PROPPATCH.

Tableau 19-2. Codes d'état des méthodes PROPFIND et PROPPATCH

Code d'état défini par les méthodes Effet

200 OK HTTP PROPFIND,

Commande PROPPATCH réussie.

207 PROPFIND WEBDAV multi-états,

PROPPATCH Lors d'une action sur une ou plusieurs ressources (ou une collection), l'état de chaque objet est encapsulé dans une réponse 207. Il s'agit d'une réponse de réussite typique.

401 HTTP PROPATCH non autorisé Nécessite une autorisation pour terminer l'opération de modification de propriété.

403 HTTP PROPFIND interdit,

Pour PROPFIND, le client n'est pas autorisé à accéder à la propriété. Pour PROPPATCH, le client ne peut pas modifier la propriété.

404 Non trouvé HTTP PROPFIND Aucune propriété de ce type.

409 Conflit HTTP PROPPATCH Conflit de sémantique de mise à jour, par exemple, tentative de mise à jour d'une propriété en lecture seule.

423 WebDAV PROPPATCH verrouillé La ressource de destination est verrouillée et il n'y a pas de jeton de verrouillage ou le jeton de verrouillage ne correspond pas.

507 Stockage insuffisant WebDAV PROPPATCH Pas assez d'espace pour enregistrer la propriété modifiée.

Gestion des collections et des espaces de noms

Une collection désigne un regroupement logique ou physique de ressources selon une hiérarchie prédéfinie. Un répertoire est un exemple classique de collection. À l'instar des répertoires d'un système de fichiers, les collections servent de conteneurs pour d'autres ressources, y compris d'autres collections (équivalentes aux répertoires du système de fichiers).

WebDAV utilise le mécanisme des espaces de noms XML. Contrairement aux espaces de noms traditionnels, les partitions d'espaces de noms XML permettent un contrôle structurel précis tout en empêchant toute modification.

collisions d'espaces de noms.

WebDAV fournit cinq méthodes pour manipuler l'espace de noms : DELETE,

MKCOL, COPY, MOVE et PROPFIND. PROPFIND a déjà été abordé dans ce chapitre, mais parlons des autres méthodes.

La méthode MKCOL

La méthode MKCOL permet aux clients de créer une collection à l'URL indiquée sur le serveur. À première vue, définir une nouvelle méthode pour créer une collection peut paraître redondant. Superposer une méthode PUT ou POST semble une alternative idéale. Les concepteurs du protocole WebDAV ont envisagé ces alternatives et ont néanmoins choisi de définir une nouvelle méthode. Voici quelques raisons qui ont motivé cette décision :

Pour qu'un PUT ou un POST crée une collection, le client doit joindre une « colle sémantique » supplémentaire à la requête. Bien que cela soit faisable, la définition d'un protocole ad hoc peut s'avérer fastidieuse et source d'erreurs.

• La plupart des mécanismes de contrôle d'accès sont basés sur le type de méthodes : seules quelques-unes sont autorisées à créer et

supprimer des ressources dans le référentiel. Si nous surchargeons d'autres méthodes, ces mécanismes de contrôle d'accès ne fonctionneront pas.

Par exemple, une demande pourrait être :

MKCOL /publication HTTP/1.1

Hôte : minstar

Contenu-Longueur : 0

Connexion : Keep-Alive

Et la réponse pourrait être :

HTTP/1.1 201 Créé

Serveur : Microsoft-IIS/5.0

Date : ven. 10 mai 2002 23:20:36 GMT

Emplacement : http://minstar/publishing/

Contenu-Longueur : 0

Examinons quelques cas pathologiques :

• Supposons que la collection existe déjà. Si une requête MKCOL /colA est effectuée et que colA existe déjà (c.-à-d. en cas de conflit d'espace de noms), la requête échouera avec le code d'état 405 « Méthode non autorisée ».

• S'il n'y a pas d'autorisations d'écriture, la demande MKCOL échouera avec un code d'état 403 For-bidden.

• Si une demande telle que MKCOL /colA/colB est effectuée et que colA n'existe pas, la demande échouera avec un code d'état de conflit 409.

Une fois le fichier ou la collection créé, nous pouvons le supprimer avec la méthode DELETE.

La méthode DELETE

Nous avons déjà vu la méthode DELETE au chapitre 3. WebDAV étend la sémantique pour couvrir les collections.

Si nous devons supprimer un répertoire, l'en-tête Depth est nécessaire. Si l'en-tête Depth n'est pas spécifié, la méthode DELETE suppose que l'en-tête Depth est défini à l'infini, c'est-à-dire que tous les fichiers du répertoire et de ses sous-répertoires sont supprimés. La réponse contient également un en-tête Content-Location identifiant la collection supprimée. La requête pourrait être la suivante :

SUPPRIMER /publier HTTP/1.0

Hôte : minstar

Et la réponse pourrait être :

HTTP/1.1 200 OK

Serveur : Microsoft-IIS/5.0

Date : mar. 14 mai 2002 16:41:44 GMT

Contenu-Emplacement : http://minstar/publishing/

Type de contenu : texte/xml

Contenu-Longueur : 0

Lors de la suppression de collections, il est toujours possible qu'un fichier soit verrouillé par quelqu'un d'autre et ne puisse être supprimé. Dans ce cas, la collection elle-même ne peut être supprimée et le serveur renvoie un code d'état 207 « Multi-Status ». La requête pourrait être la suivante :

SUPPRIMER /publier HTTP/1.0

Hôte : minstar

Et la réponse pourrait être :

HTTP/1.1 207 Multi-Statut

Serveur : Microsoft-IIS/5.0

Contenu-Emplacement : http://minstar/publishing/ .....

<?xml version="1.0"?>

<a:multistatus xmlns:a="DAV:">

<a:response>

<a:href>http://minstar/index3/ch-publish.fm</a:href>

<a:status> HTTP/1.1 423 Verrouillé </a:status>

</a:response>

</a:multistatus>

Dans cette transaction, l'élément XML <status> contient le code d'état 403 Verrouillé, indiquant que la ressource ch-publish.fm est verrouillée par un autre utilisateur.

Les méthodes COPY et MOVE

Comme pour MKCOL, il existe des alternatives à la définition de nouvelles méthodes pour les opérations COPY et MOVE. Une alternative pour la méthode COPY consiste à exécuter une requête GET sur la source, téléchargeant ainsi la ressource, puis à la renvoyer au serveur via une requête PUT. Un scénario similaire pourrait être envisagé pour MOVE (avec l'opération DELETE supplémentaire).

Cependant, ce processus n'est pas évolutif ; il faut considérer tous les problèmes liés à la gestion d'une opération COPY ou MOVE sur une collection multiniveau.

Les méthodes COPY et MOVE utilisent l'URL de la requête comme source et le contenu de l'en-tête HTTP Destination comme cible. La méthode MOVE effectue des tâches supplémentaires par rapport à la méthode COPY : elle copie l'URL source vers la destination, vérifie l'intégrité de l'URI nouvellement créée, puis supprime la source. La requête pourrait être :

{COPIER,DÉPLACER} /publication HTTP/1.1

Destination : http://minstar/pub-new

Profondeur : infini

Écraser : T

Hôte : minstar

Et la réponse pourrait être :

HTTP/1.1 201 Créé

Serveur : Microsoft-IIS/5.0

Date : mer. 15 mai 2002 18:29:53 GMT

Emplacement : http://minstar.inktomi.com/pub-new/

Type de contenu : texte/xml

Contenu-Longueur : 0

Lors d'une action sur une collection, le comportement de COPY ou MOVE est affecté par l'en-tête Depth. En l'absence d'en-tête Depth, l'infini est supposé (c'est-à-dire que, par défaut, la structure entière du répertoire source est copiée ou déplacée). Si Depth est défini à zéro, la méthode s'applique uniquement à la ressource. Si nous effectuons une copie ou un déplacement d'une collection, seule une collection ayant des propriétés identiques à celles de la source est créée à la destination ; aucun membre interne de la collection n'est copié ou déplacé.

Pour des raisons évidentes, seule une valeur de profondeur infinie est autorisée avec la méthode MOVE.

Effet d'écrasement d'en-tête

Les méthodes COPY et MOVE peuvent également utiliser l'en-tête Overwrite. Cet en-tête peut être défini sur T ou F. S'il est défini sur T et que la destination existe, une opération DELETE avec une valeur de profondeur infinie est exécutée sur la ressource de destination avant une opération COPY ou MOVE. Si l'indicateur Overwrite est défini sur F et que la ressource de destination existe, l'opération échoue.

COPIE/DÉPLACEMENT de propriétés

Lorsqu'une collection ou un élément est copié, toutes ses propriétés sont copiées par défaut. Cependant, une requête peut contenir un corps XML facultatif fournissant des informations supplémentaires pour l'opération. Vous pouvez spécifier que toutes les propriétés doivent être copiées pour que l'opération réussisse, ou définir quelles propriétés doivent être copiées pour que l'opération réussisse.

Voici quelques cas pathologiques à considérer :

• Supposons que COPY ou MOVE soit appliqué à la sortie d'un programme CGI ou d'un autre script générant du contenu. Afin de préserver la sémantique, si un fichier généré par un script CGI doit être copié ou déplacé, WebDAV fournit les éléments XML « src » et « link » qui pointent vers l'emplacement du programme ayant généré la page.

• Les méthodes COPY et MOVE peuvent ne pas être en mesure de dupliquer intégralement toutes les propriétés actives. Prenons l'exemple d'un programme CGI. S'il est copié hors du répertoire cgi-bin, il risque de ne plus être exécuté. La spécification actuelle de WebDAV fait de COPY et MOVE une solution optimale, copiant toutes les propriétés statiques et les propriétés actives appropriées.

Ressources verrouillées et COPIE/DÉPLACEMENT

Si une ressource est actuellement verrouillée, ni COPY ni MOVE ne peuvent déplacer ou dupliquer le verrou de la destination. Dans les deux cas, si la destination doit être créée sous une collection existante avec son propre verrou, la ressource dupliquée ou déplacée est ajoutée au verrou. Prenons l'exemple suivant :

COPIE /publication HTTP/1.1

Destination : http://minstar/archived/publishing-old

Supposons que /publishing et /archived soient déjà sous deux verrous différents, lock1 et lock2. Une fois l'opération COPY terminée, /publishing reste sous la portée de lock1, tandis que, suite à son déplacement vers une collection déjà verrouillée par lock2, publishing-old est ajouté à lock2. Si l'opération était un MOVE, seul publishing-old est ajouté à lock2.

Le tableau 19-3 répertorie la plupart des codes d’état possibles pour les méthodes MKCOL, DELETE, COPY et MOVE.

Tableau 19-3. Codes d'état pour les méthodes MKCOL, DELETE, COPY et MOVE

Code d'état défini par les méthodes Effet

102 Traitement WebDAV MOVE, Si la requête prend plus de 20 secondes, le serveur envoie

COPIEZ ce code d'état pour éviter toute expiration du délai d'attente des clients. Ce problème survient généralement lors de la COPIE ou du DÉPLACEMENT d'une collection volumineuse.

201 Création de HTTP MKCOL. Pour MKCOL, une collection a été créée. Pour COPIER et DÉPLACER,

COPIE, une ressource/collection a été copiée ou déplacée avec succès.

SE DÉPLACER

Tableau 19-3. Codes d'état pour les méthodes MKCOL, DELETE, COPY et MOVE (suite)

Code d'état défini par les méthodes Effet

204 Aucun contenu HTTP DELETE,

COPIE,

DÉPLACER : pour SUPPRIMER, réponse de réussite standard. Pour COPIER et DÉPLACER, la ressource a été copiée ou déplacée avec succès pour remplacer une entité existante.

207 WebDAV multi-statuts MKCOL,

COPIE,

Pour MKCOL, il s'agit d'une réponse de réussite typique. Pour COPY et MOVE, si une erreur est associée à une ressource autre que l'URI de la requête, le serveur renvoie une réponse 207 avec le corps XML détaillant l'erreur.

403 HTTP MKCOL interdit,

COPIE,

Pour MKCOL, le serveur n'autorise pas la création d'une collection à l'emplacement spécifié. Pour COPY et MOVE, la source et la destination sont identiques.

409 Conflit HTTP MKCOL,

COPIE,

DÉPLACER Dans tous les cas, les méthodes tentent de créer une collection ou une ressource lorsqu'une collection intermédiaire n'existe pas, par exemple, en essayant de créer colA/colB lorsque colA n'existe pas.

412 Échec de la précondition HTTP COPY, MOVE Soit l'en-tête Overwrite est défini sur F et la destination existe, soit le corps XML

spécifie une certaine exigence (comme la conservation de la propriété « liveness ») et les méthodes COPY ou MOVE ne sont pas en mesure de conserver la propriété.

415 Type de média non pris en charge HTTP MKCOL Le serveur ne prend pas en charge ou ne comprend pas la création du type d'entité de demande.

422 Entité non traitable WebDAV MKCOL Le serveur ne comprend pas le corps XML envoyé avec la demande.

423 WebDAV verrouillé SUPPRIMER,

COPIE,

DÉPLACER La ressource source ou de destination est verrouillée, ou le jeton de verrouillage fourni avec la méthode ne correspond pas.

502 Bad Gateway HTTP COPY, MOVE La destination est sur un serveur différent et les autorisations sont manquantes.

507 Stockage insuffisant WebDAV MKCOL

COPIE Il n'y a pas assez d'espace libre pour créer la ressource.

Méthodes HTTP/1.1 améliorées

WebDAV modifie la sémantique des méthodes HTTP DELETE, PUT et OPTIONS. La sémantique des méthodes GET et HEAD reste inchangée. Les opérations effectuées par POST sont toujours définies par l'implémentation serveur spécifique, et WebDAV ne modifie aucune sémantique POST. Nous avons déjà abordé la méthode DELETE dans la section « Gestion des collections et des espaces de noms ». Nous aborderons les méthodes PUT et OPTIONS ici.

La méthode PUT

Bien que PUT ne soit pas défini par WebDAV, c'est le seul moyen pour un auteur de transférer du contenu vers un site partagé. Nous avons abordé les fonctionnalités générales de PUT au chapitre 3. WebDAV modifie son comportement pour prendre en charge le verrouillage.

Considérez l’exemple suivant :

PUT /ch-publish.fm HTTP/1.1

Accepter: */*

Si :<http://minstar/index.htm>(<opaquelocktoken:********>)

Agent utilisateur : Client DAV (C)

Hébergeur : minstar.inktomi.com

Connexion : Keep-Alive

Contrôle du cache : pas de cache

Contenu-Longueur : 1155

Pour prendre en charge le verrouillage, WebDAV ajoute un en-tête If à la requête PUT. Dans la transaction ci-dessus, la sémantique de l'en-tête If indique que si le jeton de verrouillage spécifié correspond au verrou de la ressource (ici, ch-publish.fm), l'opération PUT doit être exécutée. L'en-tête If est également utilisé avec d'autres méthodes, telles que PROPPATCH, DELETE, MOVE, LOCK, UNLOCK, etc.

La méthode OPTIONS

Nous avons abordé la méthode OPTIONS au chapitre 3. Il s'agit généralement de la première requête d'un client WebDAV. Grâce à la méthode OPTIONS, le client tente d'établir la capacité du serveur WebDAV. Prenons une transaction dont la requête est la suivante :

OPTIONS /ch-publish.fm HTTP/1.1

Accepter: */*

Hôte : minstar.inktomi.com Et la réponse se lit comme suit :

HTTP/1.1 200 OK

Serveur : Microsoft-IIS/5.0

MS-Auteur-Via : DAV

DASL : <DAV:sql>

DAV: 1, 2

Public : OPTIONS, TRACE, GET, HEAD, DELETE, PUT, POST, COPY, MOVE, MKCOL,PROPFIND,

PROPPATCH, VERROUILLER, DÉVERROUILLER, RECHERCHER

Autoriser : OPTIONS, TRACE, GET, HEAD, DELETE, PUT, COPY, MOVE, PROPFIND, PROPPATCH,

RECHERCHER, VERROUILLER, DÉVERROUILLER

La réponse à la méthode OPTIONS comporte plusieurs en-têtes intéressants. Voici un examen légèrement désordonné :

• L'en-tête DAV contient les informations sur les classes de conformité DAV. Il existe deux classes de conformité :

Conformité de classe 1

Exige que le serveur se conforme à toutes les exigences MUST dans toutes les sections de la RFC 2518. Si la ressource est conforme uniquement au niveau de classe 1, elle enverra 1 avec l'en-tête DAV.

Conformité de classe 2

Répond à toutes les exigences de classe 1 et prend en charge la méthode LOCK. Outre LOCK, la conformité de classe 2 requiert la prise en charge des en-têtes Timeout et Lock-Token, ainsi que des éléments XML <supportedlock> et <lockdiscovery>. Une valeur de 2 dans l'en-tête DAV indique une conformité de classe 2.

Dans l'exemple ci-dessus, l'en-tête DAV indique la conformité aux classes 1 et 2.

• L'en-tête Public répertorie toutes les méthodes prises en charge par ce serveur particulier.

• L'en-tête Allow contient généralement un sous-ensemble des méthodes de l'en-tête Public. Il répertorie uniquement les méthodes autorisées sur cette ressource (ch-publish.fm).

• L'en-tête DASL fournit le type de grammaire de requête utilisé dans la méthode SEARCH. Dans ce cas, il s'agit de SQL. Plus de détails sur l'en-tête DASL sont disponibles sur http://www.webdav.org.

Gestion des versions dans WebDAV

Cela peut paraître ironique, compte tenu du « V » de « DAV », mais le contrôle de version est une fonctionnalité qui n'a pas été retenue. Dans un environnement collaboratif multi-auteurs, la gestion des versions est essentielle. En fait, pour résoudre complètement le problème de perte de mise à jour (illustré à la figure 19-3), le verrouillage et le contrôle de version sont essentiels. Parmi les fonctionnalités courantes associées au contrôle de version, on trouve la possibilité de stocker et d'accéder aux versions précédentes des documents, ainsi que la possibilité de gérer l'historique des modifications et les annotations associées détaillant ces modifications.

Le contrôle de version a été ajouté à WebDAV dans la RFC 3253.

L'avenir de WebDAV

WebDAV est aujourd'hui bien pris en charge. Parmi les implémentations clientes fonctionnelles, on trouve IE 5.x et versions ultérieures, l'Explorateur Windows et Microsoft Office. Côté serveur, on trouve notamment IIS 5.x et versions ultérieures, Apache avec mod_dav, et bien d'autres. Windows XP et Mac OS 10.x prennent en charge WebDAV dès sa sortie ; ainsi, toutes les applications conçues pour fonctionner sur ces systèmes d'exploitation sont compatibles WebDAV nativement.

Pour plus d'informations

Pour plus d'informations, reportez-vous à :

http://officeupdate.microsoft.com/frontpage/wpp/serk/

Kit de ressources des extensions serveur Microsoft FrontPage 2000.

http://www.ietf.org/rfc/rfc2518.txt?number=2518

« Extensions HTTP pour la création distribuée — WEBDAV », par Y. Goland, J.

Whitehead, A. Faizi, S. Carter et D. Jensen.

http://www.ietf.org/rfc/rfc3253.txt?number=3253

« Extensions de contrôle de version pour WebDAV », par G. Clemm, J. Amsden, T. Ellison, C.

Kaler et J. Whitehead.

http://www.ics.uci.edu/pub/ietf/webdav/intro/webdav_intro.pdf

« WEBDAV : norme IETF pour la création collaborative sur le Web », par J. Whitehead et M. Wiggins.

http://www.ics.uci.edu/~ejw/http-future/whitehead/http_pos_paper.html

« Leçons de WebDAV pour l'infrastructure Web de nouvelle génération », par J. Whitehead.

http://www.microsoft.com/msj/0699/dav/davtop.htm

« Extensions de création et de gestion de versions distribuées pour HTTP Enable Team

« Auteur », par L. Braginski et M. Powell.

http://www.webdav.org/dasl/protocol/draft-dasl-protocol-00.html

« Recherche et localisation DAV », par S. Reddy, D. Lowry, S. Reddy, R. Henderson,

J. Davis et A. Babich.

Pour plus d'informations

Chapitre 20Ceci est le titre du livreCHAPITRE 20

Redirection et équilibrage de charge

HTTP ne circule pas seul sur le Web. Les données d'un message HTTP sont régies par de nombreux protocoles tout au long de leur parcours. HTTP ne se préoccupe que des points d'extrémité du parcours (l'expéditeur et le destinataire). Cependant, dans un monde où les serveurs sont en miroir, les proxys web et les caches sont omniprésents, la destination d'un message HTTP n'est pas toujours évidente.

Ce chapitre traite des technologies de redirection : outils, techniques et protocoles réseau qui déterminent la destination finale d'un message HTTP. Les technologies de redirection déterminent généralement si le message aboutit sur un proxy, un cache ou un serveur web spécifique d'une batterie de serveurs. Elles peuvent envoyer vos messages vers des emplacements non explicitement demandés par le client.

Dans ce chapitre, nous examinerons les techniques de redirection suivantes, leur fonctionnement et leurs capacités d'équilibrage de charge (le cas échéant) :

• Redirection HTTP

• Redirection DNS

• Routage anycast

• Routage des politiques

• Transfert IP MAC

• Redirection d'adresse IP

• Le protocole de coordination du cache Web (WCCP)

• Le protocole de communication intercache (ICP)

• Le protocole de mise en cache hypertexte (HTCP)

• Le protocole de contrôle des éléments de réseau (NECP)

• Le protocole de routage de tableau de cache (CARP)

• Le protocole de découverte automatique de proxy Web (WPAD)

Pourquoi rediriger ?

La redirection est une réalité du Web moderne, car les applications HTTP veulent toujours faire trois choses :

• Effectuer des transactions HTTP de manière fiable

• Minimiser les retards

• Conserver la bande passante du réseau

Pour ces raisons, le contenu web est souvent distribué à plusieurs endroits. Ceci est fait pour des raisons de fiabilité : en cas de panne d'un emplacement, un autre est disponible ; cela permet de réduire les temps de réponse : si les clients peuvent accéder à une ressource plus proche, ils reçoivent le contenu demandé plus rapidement ; et cela permet de réduire la congestion du réseau en répartissant les serveurs cibles. La redirection peut être considérée comme un ensemble de techniques permettant de trouver le « meilleur » contenu distribué.

Le sujet de l'équilibrage de charge est abordé car la redirection et l'équilibrage de charge coexistent. La plupart des déploiements de

redirection incluent une forme d'équilibrage de charge ; ils permettent de répartir la charge des messages entrants entre plusieurs serveurs. À l'inverse, toute forme d'équilibrage de charge implique une redirection, car les messages entrants doivent être répartis d'une manière ou d'une autre entre les serveurs partageant la charge.

Où rediriger

Les serveurs, proxys, caches et passerelles sont tous perçus comme des serveurs par les clients, dans le sens où un client leur envoie une requête HTTP et qu'ils la traitent. De nombreuses techniques de redirection fonctionnent pour les serveurs, proxys, caches et passerelles en raison de leurs caractéristiques communes, similaires à celles des serveurs. D'autres techniques de redirection sont spécifiquement conçues pour une classe particulière de points de terminaison et ne sont pas d'application générale. Nous aborderons les techniques générales et spécialisées dans les sections suivantes de ce chapitre.

Les serveurs web traitent les requêtes IP par IP. Répartir les requêtes sur des serveurs dupliqués signifie que chaque requête pour une URL spécifique doit être envoyée à un serveur web optimal (le plus proche du client, le moins chargé, ou une autre optimisation). Rediriger vers un serveur revient à envoyer tous les automobilistes en quête d'essence à la station-service la plus proche.

Les proxys ont tendance à gérer les requêtes selon le protocole. Idéalement, tout le trafic HTTP à proximité d'un proxy devrait transiter par ce dernier. Par exemple, si un cache proxy est situé à proximité de plusieurs clients, toutes les requêtes transiteront idéalement par le cache proxy, car celui-ci stockera les documents les plus populaires et les servira directement, évitant ainsi des trajets plus longs et plus coûteux vers les serveurs d'origine. Rediriger vers un proxy revient à détourner le trafic d'une voie d'accès principale (quelle que soit sa destination) vers un raccourci local.

Où rediriger

Présentation des protocoles de redirection

L'objectif de la redirection est d'envoyer les messages HTTP aux serveurs web disponibles le plus rapidement possible. La direction d'un message HTTP sur Internet est affectée par les applications HTTP et les dispositifs de routage qu'il utilise. Par exemple :

• L'application de navigateur qui crée le message du client peut être configurée pour l'envoyer à un serveur proxy.

• Les résolveurs DNS choisissent l'adresse IP utilisée pour adresser le message. Cette adresse IP peut être différente pour différents clients dans différentes zones géographiques

emplacements.

• Lorsque le message traverse les réseaux, il est divisé en paquets adressés ; les commutateurs et les routeurs examinent l'adressage TCP/IP des paquets et prennent des décisions concernant le routage des paquets sur cette base.

• Les serveurs Web peuvent renvoyer les requêtes vers différents serveurs Web avec des redirections HTTP.

La configuration du navigateur, le DNS, le routage TCP/IP et HTTP fournissent tous des mécanismes de redirection des messages. Notez que certaines méthodes, comme la configuration du navigateur, ne sont utiles que pour rediriger le trafic vers des proxys, tandis que d'autres, comme la redirection DNS, peuvent être utilisées pour envoyer du trafic vers n'importe quel serveur.

Le tableau 20-1 résume les méthodes de redirection utilisées pour rediriger les messages vers les serveurs, chacune d’entre elles étant décrite plus loin dans ce chapitre.

Tableau 20-1. Méthodes générales de redirection

Mécanisme Comment ça marche Base du réacheminement Limitations

Redirection HTTP : la requête HTTP initiale est transmise à un premier serveur web qui choisit le « meilleur » serveur web pour diffuser le contenu. Le premier serveur web envoie au client une redirection HTTP vers le serveur choisi. Le client renvoie la requête au serveur choisi. De nombreuses options sont possibles, de l'équilibrage de charge à tour de rôle à la minimisation de la latence, en passant par le choix du chemin le plus court. Peut être lent : chaque transaction implique une étape de redirection supplémentaire. De plus, le premier serveur doit être capable de gérer la charge de la requête.

Redirection DNS : le serveur DNS détermine l'adresse IP à renvoyer pour le nom d'hôte dans l'URL. De nombreuses options sont possibles, de l'équilibrage de charge circulaire à la réduction de la latence, en passant par le choix du chemin le plus court. Le serveur DNS doit être configuré.

Adressage anycast : plusieurs serveurs utilisent la même adresse IP. Chaque serveur se fait passer pour un routeur principal. Les autres routeurs envoient les paquets adressés à l'adresse IP partagée au serveur le plus proche (en pensant envoyer les paquets au routeur le plus proche). Les routeurs utilisent des fonctionnalités intégrées de routage par le plus court chemin. Nécessité de posséder et de configurer des routeurs. Risques de conflits d'adresses. Les connexions

TCP établies peuvent être interrompues si le routage change et que les paquets associés à une connexion sont envoyés à différents serveurs.

Tableau 20-1. Méthodes générales de redirection (suite)

Mécanisme	Comment ça marche	Base du réacheminement	Limitations
Transfert MAC IP	Un élément réseau tel qu'un commutateur ou un routeur lit l'adresse de destination d'un paquet ; si le paquet doit être redirigé, le commutateur lui donne l'adresse MAC de destination d'un serveur ou d'un proxy.	Économisez de la bande passante et améliorez la qualité de service. Équilibrez la charge.	Le serveur ou le proxy doit être à un saut de distance.
Le commutateur de couche 4 de transfert d'adresse IP évalue le port de destination d'un paquet et remplace l'adresse IP d'un paquet redirigé par celle d'un proxy ou d'un serveur miroir.	Économisez de la bande passante et améliorez la qualité de service. Équilibrez la charge.	L'adresse IP du client peut être perdue par le serveur/proxy.	

Le tableau 20-2 résume les méthodes de redirection utilisées pour rediriger les messages vers les serveurs proxy.

Tableau 20-2. Techniques de redirection de proxy et de cache

Mécanisme	Comment ça marche	Base du réacheminement	Limitations
Configuration explicite du navigateur : le navigateur web est configuré pour envoyer des messages HTTP à un proxy proche, généralement un cache. La configuration peut être effectuée par l'utilisateur final ou par un service gérant le navigateur.	Économisez de la bande passante et améliorez la qualité de service. Équilibrez la charge.	Dépend de la capacité de configuration du navigateur.	
Configuration automatique du proxy (PAC) : le navigateur web récupère un fichier PAC depuis un serveur de configuration. Ce fichier indique au navigateur le proxy à utiliser pour chaque URL.	Économisez la bande passante et améliorez la qualité de service. Équilibrez la charge.	Le navigateur doit être configuré pour interroger le serveur de configuration.	
Proxy Web			
Découverte automatique			
Protocole WPAD : un navigateur web demande à un serveur de configuration l'URL d'un fichier PAC. Contrairement au protocole PAC seul, le navigateur n'a pas besoin d'être configuré avec un serveur de configuration spécifique. Le serveur de configuration base l'URL sur les informations contenues dans les en-têtes de requête HTTP du client.	Équilibrage de charge.	Seuls quelques navigateurs prennent en charge WPAD.	

Cache Web

Coordination

Le routeur de protocole WCCP évalue l'adresse de destination d'un paquet et encapsule les paquets redirigés avec l'adresse IP d'un proxy ou d'un serveur miroir. Compatible avec de nombreux routeurs existants. L'encapsulation des paquets permet de conserver l'adresse IP du client. Économise la bande passante et améliore la qualité de service. Équilibre la charge. L'utilisation de routeurs compatibles WCCP est obligatoire. Certaines limitations topologiques sont possibles.

Cache Internet

Protocole (ICP) : un cache proxy peut interroger un groupe de caches frères pour obtenir le contenu demandé. Il prend également en charge les hiérarchies de cache. Obtenir du contenu d'un cache frère ou parent est plus rapide que de l'appliquer au serveur d'origine. Des erreurs de cache peuvent survenir.

car seule l'URL est utilisée pour demander du contenu.

Protocole de routage de réseau de caches (CARP) : protocole de hachage de cache proxy. Il permet à un cache de transmettre une requête à un cache parent. Contrairement à ICP, le contenu des caches est disjoint et le groupe de caches agit comme un seul grand cache. Obtenir du contenu depuis un cache homologue proche est plus rapide que de l'appliquer au serveur d'origine. CARP ne prend pas en charge les relations de fratrie. Tous les clients CARP doivent s'accorder sur la configuration ; sinon, différents clients enverront le même URI à différents parents, ce qui réduira les taux de succès.

Présentation des protocoles de redirection

Tableau 20-2. Techniques de redirection de proxy et de cache (suite)

Mécanisme Comment ça marche Base du réacheminement Limitations

Mise en cache hypertexte

Protocole (HTCP) : les caches proxy participants peuvent interroger un groupe de caches frères pour obtenir le contenu demandé. Prise en charge des en-têtes HTTP 1.0 et 1.1 pour affiner les requêtes de cache. Obtenir du contenu à partir d'un cache frère ou parent est plus rapide que de l'appliquer au serveur d'origine.

Méthodes générales de redirection

Dans cette section, nous approfondirons les différentes méthodes de redirection couramment utilisées pour les serveurs et les proxys. Ces techniques permettent de rediriger le trafic vers un autre serveur (probablement plus optimal) ou de le diriger vers un proxy. Nous aborderons plus précisément la redirection HTTP, la redirection DNS,

l'adressage anycast, la redirection IP MAC et la redirection d'adresses IP.

Redirection HTTP

Les serveurs web peuvent renvoyer de courts messages de redirection aux clients, leur indiquant d'essayer ailleurs. Certains sites web utilisent la redirection HTTP comme une forme simple d'équilibrage de charge ; le serveur qui gère la redirection (le serveur de redirection) trouve le serveur de contenu le moins chargé disponible et redirige le navigateur vers ce serveur. Pour les sites web largement distribués, déterminer le « meilleur » serveur disponible devient plus complexe, prenant en compte non seulement la charge des serveurs, mais aussi la distance Internet entre le navigateur et le serveur. L'un des avantages de la redirection HTTP par rapport à d'autres formes de redirection est que le serveur de redirection connaît l'adresse IP du client ; en théorie, il peut être en mesure de faire un choix plus éclairé.

Voici comment fonctionne la redirection HTTP. Dans la figure 20-1a, Alice envoie une requête à www.joes-hardware.com :

OBTENIR /hammers.html HTTP/1.0

Hébergeur : www.joes-hardware.com

Agent utilisateur : Mozilla/4.51 [fr] (X11; U; IRIX 6.2 IP22)

Dans la figure 20-1b, au lieu de renvoyer un corps de page Web avec le code d'état HTTP 200, le serveur renvoie un message de redirection avec le code d'état 302 :

Redirection HTTP/1.0 302

Serveur : Stronghold/2.4.2 Apache/1.3.6

Emplacement : http://161.58.228.45/hammers.html

Maintenant, dans la figure 20-1c, le navigateur renvoie la requête en utilisant l'URL redirigée, cette fois vers l'hôte 161.58.228.45 :

OBTENIR /hammers.html HTTP/1.0

Hôte : 161.58.228.45

Agent utilisateur : Mozilla/4.51 [fr] (X11; U; IRIX 6.2 IP22)

Un autre client pourrait être redirigé vers un autre serveur. Dans la figure 20-1d–f, la requête de Bob est redirigée vers 161.58.228.46.

Figure 20-1. Redirection HTTP

La redirection HTTP peut vectoriser les requêtes entre les serveurs, mais elle présente plusieurs inconvénients :

• Le serveur d'origine requiert une puissance de traitement importante pour déterminer vers quel serveur rediriger. Parfois, la puissance nécessaire pour effectuer la redirection est presque aussi importante que pour afficher la page elle-même.

• Les délais d'accès des utilisateurs sont augmentés, car deux allers-retours sont nécessaires pour accéder aux pages.

• Si le serveur de redirection est cassé, le site sera cassé.

En raison de ces faiblesses, la redirection HTTP est généralement utilisée en combinaison avec certaines autres techniques de redirection.

Redirection DNS

Chaque fois qu'un client tente d'accéder au site web de Joe's Hardware, le nom de domaine www.joes-hardware.com doit être résolu en une adresse IP. Le résolveur DNS peut être le système d'exploitation du client, un serveur DNS de son réseau ou un serveur DNS plus distant. Le DNS permet d'associer plusieurs adresses IP à un même domaine, et les résolveurs DNS peuvent être configurés ou programmés pour renvoyer des adresses IP différentes. Le résolveur peut utiliser des méthodes simples (round robin) ou complexes (par exemple, vérifier la charge de plusieurs serveurs et renvoyer l'adresse IP du serveur le moins chargé).

Dans la figure 20-2, Joe gère quatre serveurs pour www.joes-hardware.com. Le serveur DNS doit choisir laquelle des quatre adresses IP renvoyer pour www.joes-hardware.com. L'algorithme de décision DNS le plus simple est un simple tourniquet.

Figure 20-2. Redirection basée sur DNS

Pour un aperçu du processus de résolution DNS, consultez la référence DNS répertoriée à la fin de ce chapitre.

DNS round robin

L'une des techniques de redirection les plus courantes est également l'une des plus simples. Le DNS round robin utilise une fonctionnalité de résolution de nom d'hôte DNS pour équilibrer la charge sur une batterie de serveurs web. Il s'agit d'une stratégie d'équilibrage de charge pure, qui ne prend en compte aucun facteur lié à la localisation du client par rapport au serveur ni à la charge actuelle de ce dernier.

Voyons ce que fait réellement CNN.com. Début mai 2000, nous avons utilisé l'outil Unix nslookup pour trouver les adresses IP associées à CNN.com. L'exemple 20-1 illustre

les résultats.*

Exemple 20-1. Adresses IP pour www.cnn.com

% nslookup www.cnn.com

Nom : cnn.com

* Résultats DNS au 7 mai 2000, résolus depuis la Californie du Nord. Les valeurs spécifiques sont susceptibles d'évoluer au fil du temps, et certains systèmes DNS renvoient des valeurs différentes selon la localisation du client.

Exemple 20-1. Adresses IP pour www.cnn.com (suite)

Adresses : 207.25.71.5, 207.25.71.6, 207.25.71.7, 207.25.71.8

207.25.71.9, 207.25.71.12, 207.25.71.20, 207.25.71.22, 207.25.71.23

207.25.71.24, 207.25.71.25, 207.25.71.26, 207.25.71.27, 207.25.71.28

207.25.71.29, 207.25.71.30, 207.25.71.82, 207.25.71.199, 207.25.71.245

207.25.71.246

Alias : www.cnn.com

Le site web www.cnn.com est en réalité une ferme de 20 adresses IP distinctes ! Chaque adresse IP peut généralement renvoyer vers un serveur physique différent.

Adresses multiples et rotation d'adresses à tour de rôle

La plupart des clients DNS utilisent uniquement la première adresse du jeu d'adresses multiples. Pour équilibrer la charge, la plupart des serveurs DNS effectuent une rotation des adresses à chaque recherche. Cette rotation est souvent appelée « round robin » DNS.

Par exemple, trois recherches DNS consécutives de www.cnn.com peuvent renvoyer des listes d'adresses IP pivotées comme celles présentées dans l'exemple 20-2.

Exemple 20-2. Listes d'adresses DNS rotatives

% nslookup www.cnn.com

Nom : cnn.com

Adresses : 207.25.71.5, 207.25.71.6, 207.25.71.7, 207.25.71.8

207.25.71.9, 207.25.71.12, 207.25.71.20, 207.25.71.22, 207.25.71.23

207.25.71.24, 207.25.71.25, 207.25.71.26, 207.25.71.27, 207.25.71.28

207.25.71.29, 207.25.71.30, 207.25.71.82, 207.25.71.199, 207.25.71.245

207.25.71.246

% nslookup www.cnn.com

Nom : cnn.com

Adresses : 207.25.71.6, 207.25.71.7, 207.25.71.8, 207.25.71.9

207.25.71.12, 207.25.71.20, 207.25.71.22, 207.25.71.23, 207.25.71.24

207.25.71.25, 207.25.71.26, 207.25.71.27, 207.25.71.28, 207.25.71.29

207.25.71.30, 207.25.71.82, 207.25.71.199, 207.25.71.245, 207.25.71.246

207.25.71.5

% nslookup www.cnn.com

Nom : cnn.com

Adresses : 207.25.71.7, 207.25.71.8, 207.25.71.9, 207.25.71.12

207.25.71.20, 207.25.71.22, 207.25.71.23, 207.25.71.24, 207.25.71.25

207.25.71.26, 207.25.71.27, 207.25.71.28, 207.25.71.29, 207.25.71.30

207.25.71.82, 207.25.71.199, 207.25.71.245, 207.25.71.246, 207.25.71.5

207.25.71.6

Dans l'exemple 20-2 :

• La première adresse de la première recherche DNS est 207.25.71.5.

• La première adresse de la deuxième recherche DNS est 207.25.71.6.

• La première adresse de la troisième recherche DNS est 207.25.71.7.

DNS round robin pour l'équilibrage de charge

Comme la plupart des clients DNS utilisent uniquement la première adresse, la rotation DNS permet de répartir la charge entre les serveurs. Si le DNS ne procédait pas à la rotation des adresses, la plupart des clients enverraient toujours la charge au premier client.

La figure 20-3 montre comment la rotation DNS round-robin agit pour équilibrer la charge :

• Lorsqu'Alice tente de se connecter à www.cnn.com, elle recherche l'adresse IP via DNS et obtient 207.25.71.5 comme première adresse IP. Alice se connecte alors au serveur web 207.25.71.5 (voir la figure 20-3c).

• Lorsque Bob tente ensuite de se connecter à www.cnn.com, il recherche également l'adresse IP via DNS, mais obtient un résultat différent car la liste d'adresses a été déplacée d'une position, suite à la précédente requête d'Alice. Bob obtient un résultat.

207.25.71.6 comme première adresse IP, et il se connecte à ce serveur dans la Figure 20-3f.

Figure 20-3. Équilibrage de charge DNS round robin entre les serveurs d'une batterie de serveurs

L'impact de la mise en cache DNS

La rotation des adresses DNS répartit la charge, car chaque recherche DNS sur un serveur obtient un ordre d'adresses différent. Cependant, cet équilibrage de charge n'est pas parfait, car les résultats de la recherche DNS peuvent être mémorisés et réutilisés par les applications, les systèmes d'exploitation et certains serveurs DNS enfants primitifs. De nombreux navigateurs web effectuent une recherche DNS pour un hôte, mais utilisent ensuite la même adresse à plusieurs reprises, afin d'éliminer le coût des recherches DNS et parce que certains serveurs préfèrent communiquer avec le même client. De plus, de nombreux systèmes d'exploitation effectuent la recherche DNS automatiquement et mettent le résultat en cache, mais n'effectuent pas de rotation des adresses. Par conséquent, le round robin DNS n'équilibre généralement pas la charge d'un seul client ; un client reste généralement bloqué sur un serveur pendant une longue période.

Cependant, même si le DNS ne répartit pas les transactions d'un client unique entre les réplicas de serveurs, il répartit correctement la charge globale de plusieurs clients. Tant qu'il existe un nombre modeste de clients ayant une demande similaire, la charge sera relativement bien répartie entre les serveurs.

Autres algorithmes de redirection basés sur DNS

Nous avons déjà expliqué comment le DNS effectue la rotation des listes d'adresses à chaque requête. Cependant, certains serveurs DNS avancés utilisent d'autres techniques pour choisir l'ordre des adresses :

Algorithmes d'équilibrage de charge

Certains serveurs DNS suivent la charge sur les serveurs Web et placent les serveurs Web les moins chargés en tête de la liste.

Algorithmes de routage de proximité

Les serveurs DNS peuvent tenter de diriger les utilisateurs vers des serveurs Web à proximité, lorsque la batterie de serveurs Web est géographiquement dispersée.

Algorithmes de masquage des défauts

Les serveurs DNS peuvent surveiller l'état du réseau et acheminer les requêtes loin des interruptions de service ou d'autres défauts.

En règle générale, le serveur DNS qui exécute des algorithmes sophistiqués de suivi de serveur est un serveur faisant autorité qui est sous le contrôle du fournisseur de contenu (voir Figure 20-4).

Plusieurs services d'hébergement distribué utilisent ce modèle de redirection DNS. L'un des inconvénients de ce modèle pour les services recherchant des serveurs à proximité est que la seule information utilisée par le serveur DNS faisant autorité pour prendre sa décision est l'adresse IP du serveur local.

Serveur DNS, pas l'adresse IP du client.

Adressage Anycast

Dans l'adressage anycast, plusieurs serveurs web dispersés géographiquement possèdent exactement la même adresse IP et s'appuient sur les capacités de routage « plus court chemin » des routeurs principaux pour envoyer les requêtes client au serveur le plus proche. Cette méthode peut notamment permettre à chaque serveur web de s'annoncer comme routeur auprès d'un routeur principal voisin. Le serveur web communique avec son routeur principal voisin via un protocole de communication. Lorsque le routeur principal reçoit des paquets destinés à l'adresse anycast, il recherche (comme d'habitude) le routeur le plus proche.

Figure 20-4. Requête DNS impliquant un serveur faisant autorité

accepte cette adresse IP. Comme le serveur s'est annoncé comme routeur pour cette adresse, le routeur principal lui enverra le paquet.

Dans la figure 20-5, trois serveurs utilisent la même adresse IP, 10.10.10.1. Le serveur de Los Angeles (LA) annonce cette adresse au routeur de Los Angeles, le serveur de New York (NY) annonce la même adresse au routeur de New York, et ainsi de suite. Les serveurs communiquent avec les routeurs via un protocole de routeur. Les routeurs acheminent automatiquement les requêtes client destinées à 10.10.10.1 vers le serveur le plus proche qui annonce cette adresse.

Dans la figure 20-5, une requête pour l'adresse IP 10.10.10.1 sera acheminée vers le serveur 3.

Figure 20-5. Adressage anycast distribué

L'adressage anycast est encore une technique expérimentale. Pour que l'adressage anycast distribué fonctionne, les serveurs doivent « parler le langage des routeurs » et ces derniers doivent être capables de gérer d'éventuels conflits d'adresses, car l'adressage Internet suppose un serveur pour une adresse. (Une mauvaise utilisation peut entraîner de graves problèmes appelés « fuites de route ».) L'adressage anycast distribué est une technologie émergente qui pourrait constituer une solution pour les fournisseurs de contenu qui contrôlent leurs propres réseaux fédérateurs.

Transfert IP MAC

Dans les réseaux Ethernet, les messages HTTP sont envoyés sous forme de paquets de données adressés. Chaque paquet possède une adresse de couche 4, composée des adresses IP source et de destination, ainsi que des numéros de port TCP ; c'est cette adresse que les périphériques de couche 4 prennent en compte. Chaque paquet possède également une adresse de couche 2, l'adresse MAC (Media Access Control), à laquelle les périphériques de couche 2 (généralement les commutateurs et les concentrateurs) prêtent attention. La fonction des périphériques de couche 2 est de recevoir les paquets avec des adresses MAC entrantes spécifiques et de les transmettre à des adresses MAC sortantes spécifiques.

Dans la figure 20-6, par exemple, le commutateur est programmé pour envoyer tout le trafic de l’adresse MAC « MAC3 » vers l’adresse MAC « MAC4 ».

Figure 20-6. Commutateur de couche 2 envoyant des requêtes client à une passerelle

Un commutateur compatible couche 4 est capable d'examiner l'adressage de couche 4 (adresses IP et numéros de port TCP) et de prendre des décisions de routage en fonction de ces informations. Par exemple, un commutateur de couche 4 pourrait envoyer tout le trafic web destiné au port 80 vers un proxy. Dans la figure 20-7, le commutateur est programmé pour envoyer tout le trafic du port 80 de MAC3 vers MAC6 (un cache proxy). Tout le reste du trafic MAC3 est dirigé vers MAC5.

En règle générale, si le contenu HTTP demandé est dans le cache et récent, le cache proxy le traite ; sinon, il envoie une requête HTTP au serveur d'origine pour le contenu, pour le compte du client. Le

commutateur transmet les requêtes du port 80 du proxy (MAC6) à la passerelle Internet (MAC5).

Les commutateurs de couche 4 prenant en charge le transfert MAC peuvent généralement transférer les requêtes vers plusieurs caches proxy et répartir la charge entre eux. De même, le trafic HTTP peut être transféré vers d'autres serveurs HTTP.

Figure 20-7. Transfert MAC à l'aide d'un commutateur de couche 4

Étant donné que la transmission d'adresses MAC s'effectue uniquement de point à point, le serveur ou le proxy doit être situé à un saut du commutateur.

Redirection d'adresse IP

Lors de la redirection d'adresses IP, un commutateur ou un autre périphérique compatible avec la couche 4 examine l'adressage TCP/IP des paquets entrants et les achemine en conséquence en modifiant l'adresse IP de destination plutôt que l'adresse MAC de destination. Un avantage par rapport à la redirection MAC est que le serveur de destination n'a pas besoin d'être à un saut ; il suffit qu'il soit situé en amont du commutateur, et le routage Internet de bout en bout de couche 3 habituel achemine le paquet au bon endroit. Ce type de redirection est également

appelé traduction d'adresses réseau (NAT).

Il existe cependant un problème : la symétrie du routage. Le commutateur qui accepte la connexion TCP entrante du client gère cette connexion ; il doit renvoyer la réponse au client sur cette connexion TCP. Par conséquent, toute réponse du serveur de destination ou du proxy doit retourner au commutateur (voir la figure 20-8).

Figure 20-8. Un commutateur effectuant une redirection IP vers un proxy de mise en cache ou un serveur web en miroir.

Il existe deux manières de contrôler le chemin de retour de la réponse :

• Remplacez l'adresse IP source du paquet par l'adresse IP du commutateur. Ainsi, quelle que soit la configuration réseau entre le commutateur et le serveur, le paquet de réponse est transmis au commutateur. On parle alors de NAT complet, où le dispositif de transfert IP traduit les adresses IP source et de destination. La figure 20-9 illustre l'effet de la NAT complet sur un datagramme TCP/IP. Par conséquent, l'adresse IP du client est inconnue du serveur web, qui pourrait la rechercher à des fins d'authentification ou de facturation, par exemple.

• Si l'adresse IP source reste celle du client, assurez-vous (d'un point de vue matériel) qu'aucune route n'existe directement du serveur vers le client (en contournant le commutateur). On parle parfois de half NAT. L'avantage est que le serveur obtient l'adresse IP du client, mais l'inconvénient est la nécessité d'un certain contrôle de l'ensemble du réseau entre le client et le serveur.

Figure 20-9. NAT complet d'un datagramme TCP/IP

Protocole de contrôle des éléments de réseau

Le protocole NECP (Network Element Control Protocol) permet aux éléments réseau (NE) (équipements tels que les routeurs et les commutateurs qui transmettent les paquets IP) de communiquer avec les éléments serveur (SE) (équipements tels que les serveurs web et les caches proxy qui traitent les requêtes de la couche application). NECP ne prend pas explicitement en charge l'équilibrage de charge ; il permet simplement à un SE d'envoyer des informations d'équilibrage de charge à un NE afin que ce dernier puisse équilibrer sa charge comme il l'entend. Comme WCCP, NECP offre plusieurs méthodes de transmission de paquets : transfert MAC, encapsulation GRE et NAT.

NECP prend en charge le principe des exceptions. Le SE peut décider qu'il ne peut pas desservir certaines adresses IP sources et les transmettre au NE. Ce dernier peut alors transmettre les requêtes de ces adresses IP au serveur d'origine.

Messages

Les messages NECP sont décrits dans le tableau 20-3.

Tableau 20-3. Messages du NECP

Message Qui l'envoie Signification

NECP_NOOP Aucune opération — ne rien faire.

NECP_INIT SE SE initie la communication avec NE. SE envoie ce message à NE après avoir ouvert une connexion TCP avec NE. SE doit savoir à quel port NE se connecter.

NECP_INIT_ACK NE Acquitte NECP_INIT.

NECP_KEEPALIVE NE ou SE Demande si le pair est vivant.

NECP_KEEPALIVE_ACK NE ou SE Répond au message de maintien en vie.

NECP_START SE SE dit « Je suis ici et prêt à accepter le trafic réseau. » Peut spécifier un port.

NECP_START_ACK NE reconnaît NECP_START.

NECP_STOP SE SE dit à NE « arrête de m'envoyer du trafic ».

NECP_STOP_ACK NE NE reconnaît l'arrêt.

NECP_EXCEPTION_ADD SE indique d'ajouter une ou plusieurs exceptions à la liste du NE. Les exceptions peuvent être basées sur l'IP source, l'IP de destination, le protocole (au-dessus de l'IP) ou le port.

NECP_EXCEPTION_ADD_ACK NE confirme EXCEPTION_ADD.

NECP_EXCEPTION_DEL SE Demande à NE de supprimer une ou plusieurs exceptions de sa liste.

NECP_EXCEPTION_DEL_ACK NE Confirme EXCEPTION_DEL.

NECP_EXCEPTION_RESET SE Demande à NE de supprimer toute la liste des exceptions.

NECP_EXCEPTION_RESET_ACK NE Confirme EXCEPTION_RESET.

NECP_EXCEPTION_QUERY SE Interroge la liste complète des exceptions de NE.

NECP_EXCEPTION_RESP NE Répond à la requête d'exception.

Méthodes de redirection de proxy

Jusqu'à présent, nous avons abordé les méthodes générales de redirection. L'accès au contenu peut également nécessiter l'utilisation de différents proxys (éventuellement pour des raisons de sécurité), ou il peut exister un cache proxy sur le réseau dont un client devrait tirer parti (car il sera probablement beaucoup plus rapide de récupérer le contenu mis en cache que d'accéder directement au serveur d'origine).

Mais comment les clients, comme les navigateurs web, savent-ils qu'ils doivent se connecter à un proxy ? Il existe trois méthodes pour le déterminer : par configuration explicite du navigateur, par configuration automatique dynamique et par interception transparente. Nous aborderons ces trois techniques dans cette section.

Un proxy peut, à son tour, rediriger les requêtes client vers un autre proxy. Par exemple, un cache proxy dont le contenu est absent peut choisir de rediriger le client vers un autre cache. Comme la réponse provient alors d'un emplacement différent de celui où le client a demandé la ressource, nous aborderons également plusieurs protocoles utilisés pour la redirection proxy-cache entre homologues : le protocole ICP (Internet Cache Protocol), le protocole ICP (Internet Cache Protocol) et le protocole ICP (Internet Cache Protocol).

Protocole de routage de tableau de cache (CARP) et protocole de mise en cache hypertexte (HTCP).

Configuration explicite du navigateur

La plupart des navigateurs peuvent être configurés pour contacter un serveur proxy pour le contenu. Un menu déroulant permet à l'utilisateur de saisir le nom, l'adresse IP et le numéro de port du proxy. Le navigateur contacte ensuite le proxy pour toutes les requêtes. Plutôt que de compter sur les utilisateurs pour configurer correctement leur navigateur pour utiliser les proxys, certains fournisseurs de services exigent qu'ils téléchargent des navigateurs préconfigurés. Ces navigateurs connaissent l'adresse du proxy à contacter.

La configuration explicite du navigateur présente deux inconvénients principaux :

Les navigateurs configurés pour utiliser des proxys ne contactent pas le serveur d'origine, même si le proxy ne répond pas. Si le proxy est en panne ou si le navigateur est mal configuré, l'utilisateur rencontre des problèmes de connectivité.

• Il est difficile d'apporter des modifications à l'architecture réseau et de les diffuser à tous les utilisateurs finaux. Si un fournisseur de services souhaite ajouter des proxys ou en retirer certains, les utilisateurs de navigateurs doivent modifier leurs paramètres de proxy.

Configuration automatique du proxy

La configuration explicite des navigateurs pour contacter des proxys spécifiques peut limiter les modifications de l'architecture réseau, car elle dépend de l'intervention des utilisateurs pour reconfigurer leurs navigateurs. Une méthodologie de configuration automatique, permettant aux navigateurs de se configurer dynamiquement pour contacter le bon serveur proxy, résout ce problème. Une telle méthodologie existe : le protocole d'autoconfiguration de proxy (PAC). PAC a été défini par Netscape et est pris en charge par Netscape Navigator.

Navigateurs Microsoft Internet Explorer.

L'idée de base du PAC est de permettre aux navigateurs de récupérer un fichier spécial, appelé fichier PAC, qui spécifie le proxy à contacter pour chaque URL. Le navigateur doit être configuré pour contacter un serveur spécifique pour le fichier PAC. Le navigateur récupère ensuite le fichier PAC à chaque redémarrage.

Le fichier PAC est un fichier JavaScript, qui doit définir la fonction :

fonction FindProxyForURL(url, hôte)

Les navigateurs appellent cette fonction pour chaque URL demandée, comme suit :

valeur_de_retour = FindProxyForURL(url_de_la_requête, hôte_dans_l'url);

La valeur de retour est une chaîne spécifiant où le navigateur doit demander cette URL. La valeur de retour peut être une liste de noms de proxys à contacter (par exemple, « PROXY proxy1.domain.com ; PROXY proxy2.domain.com ») ou la chaîne « DIRECT », ce qui signifie que le navigateur doit accéder directement au serveur d'origine, sans passer par les proxys.

La séquence d'opérations illustrant la requête et la réponse d'un navigateur pour le fichier PAC est illustrée à la figure 20-10. Dans cet exemple, le serveur renvoie un fichier PAC via un programme JavaScript. Ce programme JavaScript possède une fonction appelée « FindProxyForURL » qui indique au navigateur de contacter directement le serveur d'origine si l'hôte de l'URL demandée appartient au domaine « netscape.com », et d'accéder à « proxy1.joes-cache.com » pour toutes les autres requêtes. Le navigateur appelle cette fonction pour chaque URL demandée et se connecte en fonction des résultats renvoyés.

Figure 20-10. Configuration automatique du proxy

Le protocole PAC est très puissant : le programme JavaScript peut demander au navigateur de choisir un proxy en fonction de nombreux paramètres liés au nom d'hôte, tels que l'adresse DNS et le sous-réseau, voire le jour de la semaine ou l'heure. PAC permet aux navigateurs de contacter automatiquement le proxy approprié en cas de modification de l'architecture réseau, à condition que le fichier PAC soit mis à jour sur le serveur pour refléter les modifications d'emplacement des proxys. Le principal inconvénient de PAC est que le navigateur doit être configuré pour savoir sur quel serveur récupérer le fichier PAC ; il ne s'agit donc pas d'un système de configuration entièrement automatique. WPAD, présenté dans la section suivante, résout ce problème.

PAC, comme les navigateurs préconfigurés, est utilisé aujourd'hui par certains grands FAI.

Protocole de découverte automatique de proxy Web

Le protocole WPAD (Web Proxy Autodiscovery Protocol) vise à permettre aux navigateurs web de trouver et d'utiliser des proxys à proximité, sans que l'utilisateur final ait à configurer manuellement un proxy et sans recourir à une interception transparente du trafic. La définition d'un protocole de découverte automatique de proxy web est complexe en raison de la multitude de protocoles de découverte disponibles et des différences de configuration des proxys selon les navigateurs.

Cette section contient une version abrégée et légèrement réorganisée du projet Internet WPAD. Ce projet est actuellement en cours

d'élaboration au sein du groupe de travail sur les intermédiaires Web de l'IETF.

Découverte automatique des fichiers PAC

WPAD permet aux clients HTTP de localiser un fichier PAC et de l'utiliser pour déterminer le nom d'un serveur proxy approprié. WPAD ne détermine pas directement le nom du serveur proxy, car cela contournerait les fonctionnalités supplémentaires offertes par les fichiers PAC (équilibrage de charge, routage des requêtes vers un ensemble de serveurs, basculement automatique vers des serveurs proxy de secours, etc.).

Comme illustré à la figure 20-11, le protocole WPAD détecte l'URL d'un fichier PAC, également appelé URL de configuration (CURL). Le fichier PAC exécute un programme JavaScript qui renvoie l'adresse d'un serveur proxy approprié.

Figure 20-11. WPAD détermine l'URL PAC, qui détermine le serveur proxy.

Un client HTTP qui implémente le protocole WPAD :

• Utilise WPAD pour trouver le fichier PAC CURL

• Récupère le fichier PAC (également appelé fichier de configuration ou CFILE) correspondant au

BOUCLE

• Exécute le fichier PAC pour déterminer le serveur proxy

• Envoie des requêtes HTTP au serveur proxy renvoyé par le fichier PAC

algorithme WPAD

WPAD utilise une série de techniques de découverte de ressources pour déterminer le CURL du fichier PAC approprié. Plusieurs techniques de découverte sont spécifiées, car toutes les organisations ne peuvent pas les utiliser toutes. Les clients WPAD essaient chaque technique, une par une, jusqu'à obtenir un CURL.

La spécification WPAD actuelle définit les techniques suivantes, dans l'ordre :

• DHCP (protocole de découverte dynamique d'hôte)

• SLP (protocole de localisation de service)

• Noms d'hôtes DNS bien connus

• Enregistrements DNS SRV

• URL du service DNS dans les enregistrements TXT

Parmi ces cinq mécanismes, seules les techniques de noms d'hôtes DHCP et DNS, bien connues, sont requises pour les clients WPAD. Nous détaillerons ces mécanismes dans les sections suivantes.

Le client WPAD envoie une série de requêtes de découverte de ressources, en utilisant les mécanismes de découverte mentionnés ci-dessus, dans l'ordre. Les clients n'utilisent que les mécanismes qu'ils prennent en charge. Lorsqu'une tentative de découverte réussit, le client utilise les informations obtenues pour construire une CURL PAC.

Si un fichier PAC est récupéré avec succès sur cette CURL, le processus se termine. Dans le cas contraire, le client reprend là où il s'était arrêté dans la série prédéfinie de requêtes de découverte de ressources. Si, après avoir essayé tous les mécanismes de découverte, aucun fichier PAC n'est récupéré, le protocole WPAD échoue et le client est configuré pour n'utiliser aucun serveur proxy.

Le client essaie d'abord DHCP, puis SLP. Si aucun fichier PAC n'est récupéré, le client passe aux mécanismes DNS.

Le client parcourt plusieurs fois les méthodes DNS SRV, les noms d'hôtes connus et les enregistrements DNS TXT. À chaque fois, la requête DNS QNAME devient de moins en moins précise. Ainsi, le client peut trouver les informations de configuration les plus précises, tout en conservant des informations moins spécifiques. Chaque recherche DNS utilise le QNAME préfixé par « wpad » pour indiquer le type de ressource concerné.

demandé.

Prenons l'exemple d'un client dont le nom d'hôte est johns-desktop.development.foo.com. Voici la séquence de tentatives de découverte qu'effectuerait un client WPAD complet :

• DHCP

• Orthophoniste

• Recherche DNS sur « QNAME=wpad.development.foo.com »

• Recherche DNS SRV sur « QNAME=wpad.development.foo.com »

• Recherche DNS TXT sur « QNAME=wpad.development.foo.com »

• Recherche DNS sur « QNAME=wpad.foo.com »

• Recherche DNS SRV sur « QNAME=wpad.foo.com »

• Recherche DNS TXT sur « QNAME=wpad.foo.com »

Consultez la spécification WPAD pour obtenir un pseudo-code détaillé couvrant l'ensemble de la séquence d'opérations. Les sections suivantes décrivent les deux mécanismes requis : DHCP et la recherche DNS A. Pour plus de détails sur les méthodes de découverte CURL, consultez la spécification WPAD.

Découverte CURL à l'aide de DHCP

Pour que ce mécanisme fonctionne, les CURL doivent être stockées sur des serveurs DHCP interrogeables par les clients WPAD. Le client WPAD obtient la CURL en envoyant une requête DHCP à un serveur DHCP. La CURL est contenue dans le code d'option DHCP 252 (si le serveur DHCP est configuré avec cette information). Toutes les implémentations de clients WPAD doivent prendre en charge DHCP. Le protocole DHCP est détaillé dans la RFC 2131.

Consultez la RFC 2132 pour obtenir une liste des options DHCP existantes.

Si le client WPAD a déjà effectué des requêtes DHCP lors de son initialisation, le serveur DHCP a peut-être déjà fourni cette valeur. Si cette valeur n'est pas disponible via l'API du système d'exploitation client, le client envoie un message DHCPINFORM pour interroger le serveur DHCP.

Serveur DHCP pour obtenir la valeur.

Le code d'option DHCP 252 pour WPAD est de type STRING et de taille arbitraire.

Cette chaîne contient une URL pointant vers un fichier PAC approprié. Par exemple :

"http://server.domain/proxyconfig.pac"

Recherche d'enregistrement DNS A

Pour que ce mécanisme fonctionne, les adresses IP des serveurs proxy appropriés doivent être stockées sur des serveurs DNS interrogeables par les clients WPAD. Le client WPAD obtient l'URL CURL en envoyant une recherche d'enregistrement A à un serveur DNS. Le résultat d'une recherche réussie contient l'adresse IP d'un serveur proxy approprié.

Les implémentations client WPAD sont nécessaires pour prendre en charge ce mécanisme. Cela devrait être simple, car seule une recherche DNS de base des enregistrements A est requise. Consultez la RFC 2219 pour une description de l'utilisation d'alias DNS connus pour la découverte de ressources. Pour WPAD, la spécification utilise l'alias connu « wpad » pour la découverte automatique du proxy web.

Le client effectue la recherche DNS suivante :

QNAME=wpad.TGTDOM., QCLASS=IN, QTYPE=A

Une recherche réussie contient une adresse IP à partir de laquelle le client WPAD construit le CURL.

Récupération du fichier PAC

Une fois qu'un CURL candidat est créé, le client WPAD lui adresse généralement une requête GET. Lors de ces requêtes, les clients WPAD

doivent envoyer des en-têtes Accept contenant les informations de format CFILE appropriées qu'ils sont capables de gérer. Par exemple :

Accepter : application/x-ns-proxy-autoconfig

De plus, si le CURL entraîne une redirection, les clients sont tenus de suivre la redirection jusqu'à sa destination finale.

Quand exécuter WPAD

Le processus de découverte automatique du proxy Web doit se produire au moins aussi fréquemment que l'un des éléments suivants :

• Au démarrage du client Web, WPAD est exécuté uniquement au démarrage de la première instance. Les instances suivantes héritent des paramètres.

• Chaque fois qu'une indication provenant de la pile réseau indique que l'adresse IP de l'hôte client a changé.

Un client web peut utiliser l'une ou l'autre option, selon ce qui est pertinent dans son environnement. De plus, le client doit tenter un cycle de découverte à l'expiration d'un fichier PAC précédemment téléchargé, conformément aux règles d'expiration HTTP. Il est important que le client respecte les délais d'expiration et relance le processus WPAD à l'expiration du fichier PAC.

En option, le client peut également implémenter la réexécution du processus WPAD en cas d'échec du proxy actuellement configuré si le fichier PAC ne fournit pas d'alternative.

Lorsque le client décide d'invalider le fichier PAC actuel, il doit réexécuter l'intégralité du protocole WPAD pour s'assurer de trouver le CURL correct. Plus précisément, le protocole ne prévoit pas de récupération conditionnelle du fichier PAC (If-Modified-Since).

Plusieurs allers-retours réseau peuvent être nécessaires lors des communications de diffusion et/ou de multidiffusion du protocole WPAD. Le protocole WPAD ne doit pas être invoqué plus fréquemment que ce qui est spécifié ci-dessus (par exemple, pour la récupération par URL).

Usurpation WPAD

L'implémentation de WPAD dans IE 5 permet aux clients web de détecter automatiquement les paramètres proxy, sans intervention de l'utilisateur. L'algorithme utilisé par WPAD ajoute le nom d'hôte « wpad » au nom de domaine complet et supprime progressivement les sous-domaines jusqu'à trouver un serveur WPAD répondant au nom d'hôte ou atteindre le domaine de troisième niveau. Par exemple, les clients web du domaine abmicrosoft.com interrogeront wpad.abmicrosoft, wpad.b.microsoft.com, puis wpad.microsoft.com.

Cela a révélé une faille de sécurité, car dans les usages internationaux (et certaines autres configurations), le domaine de troisième niveau peut ne pas être fiable. Un utilisateur malveillant pourrait configurer un serveur WPAD et exécuter les commandes de configuration proxy de son choix. Les versions ultérieures d'IE (5.01 et ultérieures) ont corrigé le problème.

Délais d'attente

WPAD passe par plusieurs niveaux de découverte, et les clients doivent s'assurer que chaque phase est limitée dans le temps. Dans la mesure du possible, il est recommandé de limiter chaque phase à 10 secondes.

considéré comme raisonnable, mais les implémenteurs peuvent choisir une valeur différente, plus adaptée aux propriétés de leur réseau. Par exemple, l'implémentation d'un appareil fonctionnant sur un réseau sans fil peut utiliser un délai d'expiration beaucoup plus long pour tenir compte d'une faible bande passante ou d'une latence élevée.

Considérations de l'administrateur

Les administrateurs doivent configurer au moins une méthode de recherche d'enregistrement A DHCP ou DNS dans leurs environnements, car ce sont les deux seules que tous les clients compatibles doivent implémenter. De plus, la configuration pour prendre en charge les mécanismes plus tôt dans l'ordre de recherche améliorera le temps de démarrage du client.

L'une des principales motivations de cette structure de protocole était de prendre en charge la localisation des clients par les serveurs proxy proches. Dans de nombreux environnements, il existe plusieurs serveurs proxy (groupe de travail, passerelle d'entreprise, FAI, dorsale).

Il existe un certain nombre de points possibles auxquels des décisions de « proximité » peuvent être prises dans le cadre WPAD :

• Les serveurs DHCP de différents sous-réseaux peuvent renvoyer des réponses différentes. Ils peuvent également baser leurs décisions sur le champ cipaddr du client ou l'option d'identifiant client.

• Les serveurs DNS peuvent être configurés pour renvoyer différents enregistrements de ressources SRV/A/TXT (RR) pour différents suffixes de domaine (par exemple, QNAME wpad.marketing.bigcorp.com et wpad.development.bigcorp.com).

• Le serveur web traitant la requête CURL peut prendre des décisions en fonction de l'en-tête User-Agent, de l'en-tête Accept, de l'adresse IP/du sous-réseau/du nom d'hôte du client, de la distribution topologique des serveurs proxy à proximité, etc. Cela peut se produire dans un exécutable CGI créé pour traiter la requête CURL. Comme

mentionné précédemment, il peut même s'agir d'un serveur proxy traitant les requêtes CURL et prenant ces décisions.

• Le fichier PAC peut être suffisamment explicite pour effectuer une sélection parmi un ensemble d'alternatives lors de l'exécution sur le client. CARP repose sur ce principe pour un ensemble de caches. Il n'est pas inconcevable que le fichier PAC puisse calculer une distance réseau ou des mesures d'aptitude par rapport à un ensemble de serveurs proxy candidats, puis sélectionner le serveur le plus proche ou le plus réactif.

Méthodes de redirection du cache

Nous avons abordé les techniques de redirection du trafic vers des serveurs généraux et les techniques spécialisées pour acheminer le trafic vers des proxys et des passerelles. Cette dernière section présente quelques-unes des techniques de redirection les plus sophistiquées utilisées pour la mise en cache des serveurs proxy. Ces techniques sont plus complexes que les protocoles précédemment évoqués, car elles se veulent fiables, performantes et sensibles au contenu, en acheminant les requêtes vers des emplacements susceptibles de contenir des éléments de contenu spécifiques.

Méthodes de redirection du cache

Redirection WCCP

Cisco Systems a développé le protocole WCCP (Web Cache Coordination Protocol) pour permettre aux routeurs de rediriger le trafic web vers des caches proxy. WCCP gère la communication entre les routeurs et les caches afin que ces derniers puissent vérifier les caches (s'assurer qu'ils sont opérationnels), équilibrer la charge entre les caches et envoyer des types de trafic spécifiques à des caches spécifiques. WCCP Version 2 (WCCP2) est un protocole ouvert. Nous aborderons WCCP2 ici.

Comment fonctionne la redirection WCCP

Voici un bref aperçu du fonctionnement de la redirection WCCP pour HTTP (WCCP redirige d'autres protocoles de la même manière) :

• Commencez avec un réseau contenant des routeurs et des caches compatibles WCCP qui peuvent communiquer entre eux.

Un groupe de services WCCP est constitué d'un ensemble de routeurs et de leurs caches cibles. La configuration de ce groupe spécifie le trafic, son emplacement, son mode d'acheminement et la répartition de la charge entre les caches du groupe.

• Si le groupe de services est configuré pour rediriger le trafic HTTP, les routeurs du groupe de services envoient des requêtes HTTP aux caches du groupe de services.

• Lorsqu'une requête HTTP arrive à un routeur du groupe de services, le routeur choisit l'un des caches du groupe de services pour traiter la requête (en fonction d'un hachage sur l'adresse IP de la requête ou d'un schéma d'appariement masque/valeur).

• Le routeur envoie les paquets de requête au cache, soit en encapsulant les paquets avec l'adresse IP du cache, soit par transfert IP MAC.

• Si le cache ne peut pas répondre à la demande, les paquets sont renvoyés au routeur pour une transmission normale.

• Les membres du groupe de service échangent des messages de pulsation entre eux, vérifiant continuellement la disponibilité de chacun.

Messages WCCP2

Il existe quatre messages WCCP2, décrits dans le tableau 20-4.

Tableau 20-4. Messages WCCP2

Nom du message Qui l'envoie Informations transportées

WCCP2_HERE_I_AM Cache vers routeur Ces messages indiquent aux routeurs que des caches sont disponibles pour recevoir du trafic. Les messages contiennent toutes les informations du groupe de services du cache. Dès qu'un cache rejoint un groupe de services, il envoie ces messages à tous les routeurs du groupe. Ces messages négocient avec les routeurs qui envoient des messages WCCP2_I_SEE_YOU.

Routeur WCCP2_I_SEE_YOU vers le cache : ces messages répondent aux messages WCCP2_HERE_I_AM. Ils servent à négocier la méthode de transfert des paquets, la méthode d'affectation (qui est le cache désigné), la méthode de retour des paquets et la sécurité.

Tableau 20-4. Messages WCCP2 (suite)

Nom du message Qui l'envoie Informations transportées

WCCP2_REDIRECT_ASSIGN Cache désigné au routeur Ces messages effectuent des affectations pour l'équilibrage de charge ; ils envoient des informations de bucket pour l'équilibrage de charge de la table de hachage ou des informations de paire masque/valeur pour l'équilibrage de charge masque/valeur.

WCCP2_REMOVAL_QUERY Routeur à mettre en cache qui n'a pas envoyé de messages WCCP2_HERE_I_AM pendant 2,5 × HERE_I_AM_T secondes Si un routeur ne reçoit pas de messages WCCP2_HERE_I_AM reg-

Généralement, le routeur envoie ce message pour voir si le cache doit

être supprimé du groupe de services. La réponse appropriée d'un cache est constituée de trois messages WCCP2_HERE_I_AM identiques, séparés par HERE_I_AM_T/10 secondes.

Le format du message WCCP2_HERE_I_AM est :

En-tête de message WCCP

Composant d'informations de sécurité

Composant d'informations de service

Composant d'informations d'identité du cache Web

Composant d'informations d'affichage du cache Web

Composant d'informations sur les capacités (facultatif)

Composant d'extension de commande (facultatif)

Le format du message WCCP2_I_SEE_YOU est :

En-tête de message WCCP

Composant d'informations de sécurité

Composant d'informations de service

Composant d'informations d'identité du routeur

Composant d'informations sur la vue du routeur

Composant d'informations sur les capacités (facultatif)

Composant d'extension de commande (facultatif)

Le format du message WCCP2_REDIRECT_ASSIGN est :

En-tête de message WCCP

Composant d'informations de sécurité

Composant d'informations de service

Composant d'informations sur les devoirs ou composant de devoir alternatif

Le format du message WCCP2_REMOVAL_QUERY est :

En-tête de message WCCP

Composant d'informations de sécurité

Composant d'informations de service

Composant d'informations de requête du routeur

Composants du message

Chaque message WCCP2 est composé d'un en-tête et de composants. L'en-tête WCCP indique le type de message (Me voici, Je te vois, Affectation ou Suppression).

Requête), version WCCP et longueur du message (sans compter la longueur de l'en-tête).

Chaque composant commence par un en-tête de quatre octets décrivant son type et sa longueur. La longueur du composant n'inclut pas celle de l'en-tête. Les composants du message sont décrits dans le tableau 20-5.

Méthodes de redirection du cache

Tableau 20-5. Composants des messages WCCP2

Description du composant

Informations de sécurité : Contient l'option de sécurité et son implémentation. L'option de sécurité peut être :

WCCP2_NO_SECURITY (0)

WCCP2_MD5_SÉCURITÉ (1)

Si l'option est « aucune sécurité », le champ d'implémentation de sécurité est inexistant. Si l'option est MD5, le champ d'implémentation de sécurité est un champ de 16 octets contenant la somme de contrôle du message et le mot de passe du groupe de services. Le mot de passe ne doit pas dépasser huit octets.

Informations sur le service : décrit le groupe de services. L'ID du type de service peut avoir deux valeurs :

WCCP2_SERVICE_STANDARD (0)

WCCP2_SERVICE_DYNAMIC (1)

Si le type de service est standard, il s'agit d'un service connu, entièrement défini par son identifiant. HTTP est un exemple de service connu. Si le type de service est dynamique, les paramètres suivants définissent le service : priorité, protocole, indicateurs de service (qui déterminent le hachage) et port.

Informations sur l'identité du routeur Contient l'adresse IP et l'ID du routeur et répertorie (par adresse IP) tous les caches Web avec lesquels le routeur a l'intention de communiquer.

Informations sur l'identité du cache Web Contient l'adresse IP du cache Web et le mappage de la table de hachage de redirection.

Informations sur la vue du routeur Contient la vue du routeur du groupe de services (identités des routeurs et des caches).

Informations sur la vue du cache Web Contient la vue du cache Web du groupe de services.

Informations sur l'affectation Affiche l'affectation d'un cache Web à un bucket de hachage particulier.

Informations sur la requête du routeur Contient l'adresse IP du routeur, l'adresse du cache Web interrogé et l'ID du dernier routeur du groupe de services qui a reçu un message Here I Am du cache Web.

Informations sur les capacités Utilisées par les routeurs pour annoncer les méthodes de transfert de paquets, d'équilibrage de charge et de retour de paquets prises en charge ; utilisées par les caches Web pour indiquer aux routeurs quelle méthode le cache Web préfère.

Affectation alternative Contient des informations d'affectation de table de hachage pour l'équilibrage de charge.

Carte d'affectation Contient des éléments de masque/ensemble de valeurs pour le groupe de services.

Extension de commande utilisée par les caches Web pour indiquer aux routeurs qu'ils s'arrêtent ; utilisée par les routeurs pour reconnaître un arrêt du cache.

Groupes de services

Un groupe de services est constitué d'un ensemble de routeurs et de caches compatibles WCCP qui échangent des messages WCCP. Les routeurs envoient le trafic web aux caches du groupe de services. La configuration du groupe de services détermine la répartition du trafic entre les caches. Les routeurs et les caches échangent les informations de configuration du groupe de services dans les messages « Here I Am » et « I See You ».

Encapsulation de paquets GRE

Les routeurs prenant en charge WCCP redirigent les paquets HTTP vers un serveur spécifique en les encapsulant avec l'adresse IP de ce serveur. L'encapsulation du paquet contient également un champ proto d'en-tête IP indiquant l'encapsulation générique du routeur (GRE). L'existence de ce champ proto indique au proxy récepteur qu'il dispose d'un paquet encapsulé.

Comme le paquet est encapsulé, l'adresse IP du client n'est pas perdue. La figure 20-12 illustre l'encapsulation d'un paquet GRE.

Figure 20-12. Comment un routeur WCCP modifie l'adresse IP de destination d'un paquet HTTP

Équilibrage de charge WCCP

Outre le routage, les routeurs WCCP peuvent répartir la charge entre plusieurs serveurs de réception. Les routeurs WCCP et leurs serveurs de réception échangent des messages de pulsation pour s'informer mutuellement de leur bon fonctionnement. Si un serveur de réception cesse d'envoyer des messages de pulsation, le routeur WCCP envoie le trafic de requêtes directement à Internet, au lieu de le rediriger vers ce nœud. Lorsque le nœud est remis en service, le routeur WCCP recommence à recevoir des messages de pulsation et à lui envoyer du trafic de requêtes.

Protocole de cache Internet

Le protocole ICP (Internet Cache Protocol) permet aux caches de rechercher des correspondances de contenu dans les caches frères. Si un cache ne contient pas le contenu demandé dans un message HTTP, il peut déterminer si le contenu se trouve dans un cache frère proche et, le cas échéant, le récupérer à partir de celui-ci, évitant ainsi une requête plus coûteuse au serveur d'origine. ICP peut être considéré comme un protocole de clustering de caches. Il s'agit d'un protocole de redirection dans le sens où la destination finale d'une requête HTTP peut être déterminée par une série de requêtes ICP.

ICP est un protocole de découverte d'objets. Il interroge simultanément les caches voisins pour savoir si l'un d'eux contient une URL particulière. Les caches voisins renvoient un court message indiquant « HIT » s'ils contiennent cette URL ou « MISS » dans le cas contraire. Le cache est alors libre d'ouvrir une connexion HTTP avec un cache voisin contenant l'objet.

Le protocole ICP est simple et léger. Les messages ICP sont des structures de 32 bits compressées dans l'ordre des octets du réseau, ce qui facilite leur analyse. Ils sont transportés dans des datagrammes UDP pour

Protocole de cache Internet

efficacité. UDP est un protocole Internet peu fiable, ce qui signifie que les données peuvent être détruites pendant le transit. Les programmes qui parlent ICP doivent donc disposer de délais d'attente pour détecter les datagrammes perdus.

Voici une brève description des parties d’un message ICP :

Opcode

L'opcode est une valeur de 8 bits qui décrit la signification du message ICP. Les opcodes de base sont les messages de requête ICP_OP_QUERY et les messages de réponse ICP_OP_HIT et ICP_OP_MISS.

Version

Le numéro de version 8 bits décrit le numéro de version du protocole ICP. La version d'ICP utilisée par Squid, documentée dans la RFC Internet 2186, est la version 2.

Longueur du message

Taille totale en octets du message ICP. Comme il ne comporte que 16 bits, la taille du message ICP ne peut pas dépasser 16 383 octets. Les URL font généralement moins de 16 Ko ; au-delà, de nombreuses applications web ne les traiteront pas.

Numéro de demande

Les caches compatibles ICP utilisent le numéro de requête pour suivre les requêtes et réponses simultanées. Un message de réponse ICP doit toujours contenir le même numéro de requête que celui qui a déclenché la réponse.

Options

Le champ d'options ICP 32 bits est un vecteur binaire contenant des indicateurs qui modifient le comportement d'ICP. ICPv2 définit deux indicateurs, qui modifient tous deux les requêtes ICP_OP_QUERY. L'indicateur ICP_FLAG_HIT_OBJ active et désactive le renvoi des données de document dans les réponses ICP. L'indicateur ICP_FLAG_SRC_RTT demande une estimation du temps d'aller-retour vers le serveur d'origine, mesuré par un cache frère.

Données d'option

Les données d'option 32 bits sont réservées aux fonctionnalités facultatives. ICPv2 utilise les 16 bits de poids faible des données d'option pour conserver une estimation facultative du temps d'aller-retour entre le serveur frère et le serveur d'origine.

Adresse de l'hôte de l'expéditeur

Un champ historique contenant l'adresse IP 32 bits de l'expéditeur du message ; non utilisé dans la pratique.

Charge utile

Le contenu de la charge utile varie selon le type de message. Pour ICP_OP_QUERY, la charge utile est une adresse d'hôte du demandeur d'origine sur 4 octets, suivie d'une URL terminée par un caractère NUL. Pour ICP_OP_HIT_OBJ, la charge utile est une URL terminée par un caractère NUL.

URL suivie d'une taille d'objet de 16 bits, suivie des données de l'objet.

Pour plus d'informations sur ICP, reportez-vous aux RFC informatifs 2186 et 2187. D'excellentes références sur ICP et le peering sont également disponibles auprès du Laboratoire national américain pour la recherche appliquée sur les réseaux (http://www.nlanr.net/Squid/).

Protocole de routage de tableau de cache

Les serveurs proxy réduisent considérablement le trafic Internet en interceptant les requêtes des utilisateurs individuels et en fournissant des copies en cache des objets web demandés. Cependant, avec l'augmentation du nombre d'utilisateurs, un volume de trafic important peut surcharger les serveurs proxy eux-mêmes.

Une solution à ce problème consiste à utiliser plusieurs serveurs proxy pour répartir la charge sur un ensemble de serveurs. Le protocole CARP (Cache Array Routing Protocol) est une norme proposée par Microsoft Corporation et Netscape Communication Corporation pour administrer un ensemble de serveurs proxy de telle sorte qu'un ensemble de serveurs proxy apparaisse aux clients comme un seul cache logique.

CARP est une alternative à ICP. CARP et ICP permettent tous deux aux administrateurs d'améliorer les performances en utilisant plusieurs serveurs proxy. Cette section explique en quoi CARP diffère d'ICP, les avantages et les inconvénients de son utilisation par rapport à ICP, ainsi que les détails techniques de sa mise en œuvre.

En cas d'échec de cache dans ICP, le serveur proxy interroge les caches voisins à l'aide d'un message ICP afin de déterminer la disponibilité de l'objet web. Les caches voisins répondent par un « HIT » ou un « MISS », et le serveur proxy demandeur utilise ces réponses pour sélectionner l'emplacement le plus approprié pour récupérer l'objet. Si les serveurs proxy ICP étaient organisés hiérarchiquement, un échec serait élevé au parent. La figure 20-13 illustre schématiquement la résolution des échecs et des succès avec ICP.

Figure 20-13. Requêtes ICP

Protocole de routage de tableau de cache

Notez que chaque serveur proxy, connecté via le protocole ICP, est un serveur de cache autonome avec des miroirs de contenu redondants, ce qui signifie que des entrées d'objets web en double entre les serveurs proxy sont possibles. En revanche, l'ensemble des serveurs connectés via CARP fonctionne comme un seul et même serveur, chaque composant ne contenant qu'une fraction du total des documents mis en cache. En appliquant une fonction de hachage à l'URL d'un objet web, CARP associe les objets web à un serveur proxy spécifique. Chaque objet web ayant un répertoire d'origine unique, nous pouvons déterminer son emplacement par une seule recherche, plutôt que d'interroger chacun des serveurs proxy configurés dans l'ensemble. La figure 20-14 résume l'approche CARP.

Figure 20-14. Redirection CARP

Bien que la figure 20-14 présente le proxy de mise en cache comme l'intermédiaire entre les clients et les serveurs proxy, répartissant la charge entre ces derniers, cette fonction peut être assurée par les clients eux-mêmes. Les navigateurs commerciaux tels qu'Internet Explorer et Netscape Navigator peuvent être configurés pour calculer la fonction de hachage sous la forme d'un plug-in qui détermine le serveur proxy auquel la requête doit être envoyée.

La résolution déterministe du serveur proxy dans CARP évite d'envoyer des requêtes à tous les voisins, ce qui réduit le nombre de messages inter-cache à envoyer. À mesure que de nouveaux serveurs proxy sont ajoutés à la configuration, le système de cache collectif s'adapte assez bien. Cependant, CARP présente l'inconvénient que si l'un des serveurs proxy devient indisponible, la fonction de hachage doit être modifiée pour refléter ce changement, et le contenu des serveurs proxy doit être redistribué entre les serveurs proxy existants. Cette opération peut s'avérer coûteuse en cas de pannes fréquentes du serveur proxy. En revanche, la redondance du contenu des serveurs proxy ICP rend le redistribution superflue. Autre problème potentiel : CARP étant un nouveau protocole, les serveurs proxy existants exécutant uniquement le protocole ICP peuvent ne pas être facilement inclus dans une collection CARP.

Après avoir décrit la différence entre CARP et ICP, décrivons maintenant CARP plus en détail. La méthode de redirection CARP implique les tâches suivantes :

• Tenir un tableau des serveurs proxy participants. Ces serveurs sont interrogés régulièrement pour déterminer lesquels sont toujours actifs.

• Pour chaque serveur proxy participant, calculez une fonction de hachage. La valeur renvoyée par la fonction de hachage prend en compte la charge que ce proxy peut gérer.

• Définissez une fonction de hachage distincte qui renvoie un nombre basé sur l’URL de l’objet Web demandé.

• Calculez la somme de la fonction de hachage de l'URL et de celle des serveurs proxy pour obtenir un tableau de nombres. La valeur maximale de ces nombres détermine le serveur proxy à utiliser pour l'URL. Les valeurs calculées étant déterministes, les requêtes ultérieures pour le même objet web seront transmises au même serveur proxy.

Ces quatre tâches peuvent être effectuées soit sur le navigateur, soit dans un plug-in, soit être calculées sur un serveur intermédiaire.

Pour chaque ensemble de serveurs proxy, créez une table répertoriant tous les serveurs de l'ensemble. Chaque entrée de la table doit contenir des informations sur les facteurs de charge, les valeurs de compte à rebours de durée de vie (TTL) et les paramètres globaux, tels

que la fréquence d'interrogation des membres. Le facteur de charge indique la charge supportée par la machine, qui dépend de la vitesse du processeur et de la capacité du disque dur. La table peut être gérée à distance via une interface RPC. Une fois les champs mis à jour par RPC, ils peuvent être mis à disposition ou publiés pour les clients et les proxys en aval. Cette publication s'effectue en HTTP, permettant à tout client ou serveur proxy d'accéder aux informations de la table sans introduire de protocole inter-proxy. Les clients et les serveurs proxy utilisent simplement une URL connue pour récupérer la table.

La fonction de hachage utilisée doit garantir la répartition statistique des objets web entre les serveurs proxy participants. Le facteur de charge du serveur proxy doit être utilisé pour déterminer la probabilité statistique qu'un objet web lui soit attribué.

En résumé, le protocole CARP permet de considérer un groupe de serveurs proxy comme un cache collectif unique, plutôt que comme un groupe de caches coopérants mais distincts (comme dans ICP). Un chemin de résolution de requête déterministe trouve l'emplacement d'un objet web spécifique en un seul saut. Cela élimine le trafic inter-proxy souvent généré.

Protocole de routage de tableau de cache

Trouver l'objet web dans un groupe de serveurs proxy dans ICP. CARP évite également le stockage en double d'objets web sur différents serveurs proxy. Cela présente l'avantage de permettre au système de cache de disposer d'une plus grande capacité de stockage, mais aussi l'inconvénient : la défaillance d'un proxy nécessite le réacheminement d'une partie du contenu du cache vers les proxys existants.

Protocole de mise en cache hypertexte

Nous avons précédemment présenté ICP, un protocole permettant aux caches proxy d'interroger leurs homologues sur la présence de documents. Cependant, ICP a été conçu pour HTTP/0.9 et permet donc aux caches d'envoyer uniquement l'URL lors de l'interrogation d'un homologue sur la présence d'une ressource. Les versions 1.0 et 1.1 de HTTP ont introduit de nombreux nouveaux en-têtes de requête qui, avec l'URL, sont utilisés pour prendre des décisions concernant la correspondance des documents. Par conséquent, le simple envoi de l'URL dans une requête peut ne pas produire de réponses précises.

Le protocole de mise en cache hypertexte (HTCP) réduit le risque de fausses correspondances en permettant aux caches frères de s'interroger mutuellement sur la présence de documents à l'aide de l'URL et de tous les en-têtes de requête et de réponse. De plus, le protocole HTCP permet aux caches frères de surveiller et de demander l'ajout et la suppression de documents sélectionnés dans leurs caches respectifs, ainsi que de modifier les politiques de mise en cache de leurs documents respectifs.

La figure 20-13, qui illustre une transaction ICP, peut également servir à illustrer une transaction HTCP. Ce protocole est un autre protocole de découverte d'objets. Si un cache proche contient le document, le cache demandeur peut ouvrir une connexion HTTP vers ce cache pour en obtenir une copie. La différence entre une transaction ICP et une transaction HTCP réside dans le niveau de détail des requêtes et des réponses.

La structure des messages HTCP est illustrée à la figure 20-15. La partie « En-tête » indique la longueur et les versions du message. La partie « Données » commence par la longueur des données et inclut les opcodes, les codes de réponse, ainsi que certains indicateurs et identifiants, et se termine par les données elles-mêmes. Une section « Authentification » facultative peut suivre la section « Données ».

Les détails des champs de message sont les suivants :

En-tête

La section En-tête comprend un message de 32 bits, une version majeure du protocole de 8 bits et une version mineure du protocole de 8 bits. La longueur du message inclut la taille de l'en-tête, des données et de l'authentification.

Données

La section Données contient le message HTCP et sa structure est illustrée à la figure 20-15. Les composants des données sont décrits dans le tableau 20-6.

Figure 20-15. Format de message HTCP

Tableau 20-6. Composantes des données HTCP

Description du composant

Longueur des données Une valeur de 16 bits du nombre d'octets dans la section Données, y compris la longueur du champ Longueur lui-même.

Opcode : code d'opération à 4 bits pour la transaction HTCP. La liste complète des opcodes est fournie dans le tableau 20-7.

Code de réponse : clé de 4 bits indiquant la réussite ou l’échec de la transaction. Les valeurs possibles sont :

• 0—L'authentification n'a pas été utilisée, mais elle est nécessaire

• 1—L’authentification a été utilisée, mais elle n’est pas satisfaisante

• 2—Opcode non implémenté

• 3—Version majeure non prise en charge

• 4—Version mineure non prise en charge

• 5—Opcode inapproprié, interdit ou indésirable

F1 F1 est surchargé : si le message est une demande, F1 est un indicateur de 1 bit défini par le demandeur indiquant qu'il a besoin d'une réponse (F1=1) ; si le message est une réponse, F1 est un indicateur de 1 bit indiquant si la réponse doit être interprétée comme une réponse au message global (F1=1) ou simplement comme une réponse aux champs de données Opcode (F1=0).

RR Un indicateur de 1 bit indiquant que le message est une demande (RR=0) ou une réponse (RR=1).

ID de transaction Une valeur de 32 bits qui, combinée à l'adresse réseau du demandeur, identifie de manière unique la transaction HTCP.

Données d'opcode Les données d'opcode dépendent de l'opcode. Voir le tableau 20-7.

Protocole de mise en cache hypertexte

Le tableau 20-7 répertorie les opcodes HTCP et leurs types de données correspondants.

Tableau 20-7. Codes d'opération HTCP

Description de la valeur du code d'opération Codes de réponse Données du code d'opération

NOP 0 Essentiellement une opération « ping ». Toujours 0 Aucun

TST 1 0 si l'entité est présente, 1 si l'entité n'est pas présente Contient l'URL et les en-têtes de requête dans la requête et uniquement les en-têtes de réponse dans la réponse

LUN 2 0 si accepté, 1 si refusé

SET 3 : Le message SET permet aux caches de demander des modifications des politiques de mise en cache. Voir le tableau 20-9 pour la liste des en-têtes utilisables dans les messages SET. 0 si accepté, 1 si ignoré.

CLR 4 0 si je l'avais, mais il a maintenant disparu ;

1 si je l'avais, mais je le garde ; et 2 si je ne l'avais pas

Authentification HTCP

La partie authentification du message HTCP est facultative. Sa structure est illustrée à la figure 20-15 et ses composants sont décrits dans le tableau 20-8.

Tableau 20-8. Composants d'authentification HTCP

Description du composant

Longueur d'authentification Le nombre de 16 bits d'octets dans la section Authentification du message, y compris la longueur du champ Longueur lui-même.

Heure de signature Un nombre de 32 bits représentant le nombre de secondes depuis 00:00:00 le 1er janvier 1970 GMT au moment où la signature est générée.

Sig expire Un nombre de 32 bits représentant le nombre de secondes depuis 00:00:00 le 1er janvier 1970 GMT lorsque la signature expirera.

Nom de clé : une chaîne qui spécifie le nom du secret partagé. La section Clé comporte deux parties : la longueur de 16 bits en octets de la chaîne qui suit, suivie du flux d'octets ininterrompu de la chaîne.

Signature : le condensé HMAC-MD5 avec une valeur B de 64 (représentant les adresses IP et les ports source et de destination), les versions HTCP majeure et mineure du message, les valeurs d'heure et d'expiration de la signature, les données HTCP complètes et la clé. La signature comporte également deux parties : la longueur de la chaîne (16 bits en octets), suivie de la chaîne elle-même.

Définition des politiques de mise en cache

Le message SET permet aux caches de demander des modifications aux politiques de mise en cache des documents. Les en-têtes utilisables dans les messages SET sont décrits dans le tableau 20-9.

Tableau 20-9. Liste des en-têtes de cache pour la modification des politiques de mise en cache

Description de l'en-tête

Cache-Vary : le demandeur a appris que le contenu varie selon un ensemble d'en-têtes différent de celui de l'en-tête Vary de la réponse. Cet en-tête remplace l'en-tête Vary de la réponse.

Cache-Location La liste des caches proxy qui peuvent également avoir des copies de cet objet.

Cache-Policy : le demandeur a appris les politiques de mise en cache de cet objet plus en détail que ce qui est spécifié dans les en-têtes de réponse. Les valeurs possibles sont : « no-cache », ce qui signifie que la réponse ne peut pas être mise en cache, mais peut être partagée entre plusieurs demandeurs simultanés ; « no-share », ce qui signifie que l'objet n'est pas partageable ; et « no-cache-cookie », ce qui signifie que le contenu peut changer en raison des cookies et que la mise en cache est donc déconseillée.

Cache-Flags Le demandeur a modifié les politiques de mise en cache de l'objet et l'objet peut devoir être traité de manière spéciale et pas nécessairement conformément aux politiques réelles de l'objet.

Cache-Expiry L'heure d'expiration réelle du document telle qu'elle a été apprise par le demandeur.

Cache-MD5 La somme de contrôle MD5 calculée par le demandeur de l'objet, qui peut être différente de la valeur dans le

En-tête Content-MD5, ou peut être fourni car l'objet n'a pas d'en-tête Content-MD5.

Cache vers origine : le temps d'aller-retour mesuré par le demandeur vers un serveur d'origine. Le format des valeurs de cet en-tête est le suivant : <nom ou adresse IP du serveur d'origine> <temps d'aller-retour moyen en secondes> <nombre d'échantillons> <nombre de sauts de routeur entre le demandeur et le serveur d'origine>.

En autorisant l'envoi des en-têtes de requête et de réponse dans les messages de requête aux caches frères, HTCP peut réduire le taux de faux résultats dans les requêtes de cache. En permettant également aux caches frères d'échanger des informations de politique, HTCP peut améliorer leur capacité à coopérer.

Pour plus d'informations

Pour plus d'informations, consultez les références suivantes :

DNS et Bind

Cricket Liu, Paul Albitz et Mike Loukides, O'Reilly & Associates, Inc.

http://www.wrec.org/Drafts/draft-cooper-webi-wpad-00.txt

Protocole de découverte automatique de proxy Web.
http://home.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html

« Format de fichier de configuration automatique du proxy Navigator ».
http://www.ietf.org/rfc/rfc2186.txt

IETF RFC 2186, « Protocole de communication intercache (ICP) version 2 », par D.

Wessels et K. Claffy.

http://icp.ircache.net/carp.txt

« Protocole de routage de matrice de cache v1.0 »
http://www.ietf.org/rfc/rfc2756.txt

IETF RFC 2756, « Protocole de mise en cache hypertexte (HTCP/0.0) », par P. Vixie et D.

Wessels.

Pour plus d'informations

http://www.ietf.org/internet-drafts/draft-wilson-wrec-wccp-v2-00.txt draft-wilson-wrec-wccp-v2-01.txt, « Protocole de communication de cache Web V2.0 », par M. Cieslak, D. Forster, G. Tiwana et R. Wilson.

http://www.ietf.org/rfc/rfc2131.txt?number=2131

« Protocole de configuration dynamique d'hôte ».
http://www.ietf.org/rfc/rfc2132.txt?number=2132

Options DHCP et extensions de fournisseur BOOTP.
http://www.ietf.org/rfc/rfc2608.txt?number=2608

Protocole de localisation de service, version 2.
http://www.ietf.org/rfc/rfc2219.txt?number=2219

« Utilisation des alias DNS pour les services réseau. »

Chapitre 21uCeci est le titre du livre CHAPITRE 21

Journalisation et suivi de l'utilisation

Presque tous les serveurs et proxys enregistrent des résumés des transactions HTTP qu'ils traitent. Cela est effectué pour diverses raisons : suivi de l'utilisation, sécurité, facturation, détection des erreurs, etc. Dans ce chapitre, nous présentons brièvement la journalisation, en examinant les informations généralement enregistrées sur les transactions HTTP et les formats de journalisation courants.

Que faut-il enregistrer ?

La journalisation a généralement deux objectifs : détecter les problèmes sur le serveur ou le proxy (par exemple, identifier les requêtes qui échouent) et générer des statistiques sur les accès aux sites web. Ces statistiques sont utiles pour le marketing, la facturation et la planification des capacités (par exemple, pour déterminer les besoins en serveurs ou en bande passante supplémentaires).

Vous pourriez enregistrer tous les en-têtes d'une transaction HTTP, mais pour les serveurs et les proxys qui traitent des millions de transactions par jour, la masse de ces données deviendrait rapidement incontrôlable. Vous finiriez également par enregistrer une quantité importante d'informations qui ne vous intéressent pas vraiment et que vous ne consulteriez peut-être jamais.

En général, seuls les éléments essentiels d'une transaction sont enregistrés. Voici quelques exemples de champs fréquemment enregistrés :

• Méthode HTTP

• Version HTTP du client et du serveur

• URL de la ressource demandée

• Code d'état HTTP de la réponse

• Taille des messages de demande et de réponse (y compris les corps d'entité)

• Horodatage du moment où la transaction a eu lieu

• Valeurs d'en-tête Referer et User-Agent

La méthode HTTP et l'URL indiquent l'objectif de la requête, par exemple, obtenir une ressource ou publier un bon de commande. L'URL peut être utilisée pour suivre la popularité des pages du site web.

Les chaînes de version fournissent des indications sur le client et le serveur, utiles pour déboguer des interactions étranges ou inattendues entre clients et serveurs. Par exemple, si les requêtes échouent plus souvent que prévu, les informations de version peuvent indiquer qu'une nouvelle version d'un navigateur est incapable d'interagir avec le serveur.

Le code d'état HTTP indique ce qui est arrivé à la requête : si elle a réussi, si la tentative d'autorisation a échoué, si la ressource a été trouvée, etc. (Voir « Codes d'état » au chapitre 3 pour une liste des codes d'état HTTP.)

La taille de la requête/réponse et l'horodatage sont principalement utilisés à des fins comptables, c'est-à-dire pour suivre le nombre d'octets entrants, sortants ou traversant l'application. L'horodatage permet également de corréler les problèmes observés avec les requêtes en cours.

Formats de journaux

Plusieurs formats de journaux sont devenus la norme, et nous aborderons les plus courants dans cette section. La plupart des applications HTTP commerciales et open source prennent en charge la journalisation dans un ou plusieurs de ces formats courants. Nombre de ces applications permettent également aux administrateurs de configurer les formats de journaux et de créer leurs propres formats personnalisés.

L'un des principaux avantages de la prise en charge (pour les applications) et de l'utilisation (pour les administrateurs) de ces formats plus standard réside dans la possibilité d'exploiter les outils conçus pour traiter et générer des statistiques de base à partir de ces journaux. De nombreux packages open source et commerciaux permettent d'analyser les journaux à des fins de reporting. Grâce à l'utilisation de formats standard, les applications et leurs administrateurs peuvent exploiter ces ressources.

Format de journal commun

L'un des formats de journaux les plus courants aujourd'hui est appelé, à juste titre, « Format de journal commun ». Initialement défini par le NCSA, de nombreux serveurs l'utilisent par défaut. La plupart des serveurs commerciaux et open source peuvent être configurés pour utiliser ce format, et de nombreux outils commerciaux et gratuits permettent d'analyser les fichiers journaux courants. Le tableau 21-1 répertorie, par ordre chronologique, les champs du Format de journal commun.

Tableau 21-1. Champs communs du format de journal

Description du champ

remotehost Le nom d'hôte ou l'adresse IP de la machine du demandeur (IP si le serveur n'a pas été configuré pour effectuer un DNS inversé ou ne peut pas rechercher le nom d'hôte du demandeur)

nom d'utilisateur Si une recherche d'identité a été effectuée, le nom d'utilisateur authentifié du demandeur

Tableau 21-1. Champs du format de journal commun (suite)

Description du champ

auth-username Si l'authentification a été effectuée, le nom d'utilisateur avec lequel le demandeur s'est authentifié

horodatage La date et l'heure de la demande

request-line Le texte exact de la ligne de requête HTTP, « GET /index.html HTTP/1.1 »

code de réponse Le code d'état HTTP qui a été renvoyé dans la réponse

taille de la réponse La longueur du contenu de l'entité de réponse : si aucune entité n'a été renvoyée dans la réponse, un zéro est enregistré

La RFC 931 décrit la recherche d'identité utilisée dans cette authentification. Le protocole d'identification a été abordé au chapitre 5.

L'exemple 21-1 répertorie quelques exemples d'entrées au format de journal commun.

Exemple 21-1. Format de journal commun

209.1.32.44 - - [03/Oct/1999:14:16:00 -0400] "GET / HTTP/1.0" 200 1024 http-guide.com - dg [03/Oct/1999:14:16:32 -0400] "GET / HTTP/1.0" 200 477 http-guide.com - dg [03/Oct/1999:14:16:32 -0400] "GET /foo HTTP/1.0" 404 0 Dans ces exemples, les champs sont attribués comme suit :

Champ Entrée 1 Entrée 2 Entrée 2

hôte distant 209.1.32.44 http-guide.com http-guide.com

nom d'utilisateur <vide> <vide> <vide>

auth-username <vide> dg dg

horodatage 03/oct./1999:14:16:00 -0400 03/oct./1999:14:16:32 -0400 03/oct./1999:14:16:32 -0400

ligne de requête GET / HTTP/1.0 GET / HTTP/1.0 GET /foo HTTP/1.0

code de réponse 200 200 404

taille de la réponse 1024 477 0

Notez que le champ remotehost peut être soit un nom d'hôte, comme dans http-guide.com, soit une adresse IP, telle que 209.1.32.44.

Les tirets dans les deuxième et troisième champs (nom d'utilisateur) indiquent qu'ils sont vides. Cela signifie soit qu'aucune recherche d'identité n'a eu lieu (deuxième champ vide), soit que l'authentification n'a pas été effectuée (troisième champ vide).

Format de journal combiné

Un autre format de journal couramment utilisé est le format de journal combiné. Ce format est pris en charge par des serveurs comme Apache. Le format de journal combiné est très similaire au format de journal commun ; il le reproduit même, avec l'ajout de deux champs (listés dans le tableau 21-2). Le champ « User-Agent » permet d'identifier les applications clientes HTTP à l'origine des requêtes enregistrées, tandis que le champ « Referer » fournit plus de détails sur l'emplacement où le demandeur a trouvé cette URL.

Tableau 21-2. Champs supplémentaires du format de journal combiné

Description du champ

Référent Le contenu de l'en-tête HTTP Referer

User-Agent Le contenu de l'en-tête HTTP User-Agent

L'exemple 21-2 donne un exemple d'entrée de format de journal combiné.

Exemple 21-2. Format de journal combiné

209.1.32.44 - - [03/oct./1999:14:16:00 -0400] "GET / HTTP/1.0" 200 1024 "http://www.joeshardware.com/" "5.0 : Mozilla/4.0 (compatible ; MSIE 5.0 ; Windows 98)"

Dans l'exemple 21-2, les champs Référent et Agent utilisateur sont attribués comme suit :

Valeur du champ

Référent http://www.joes-hardware.com/

Agent utilisateur 5.0 : Mozilla/4.0 (compatible ; MSIE 5.0 ; Windows 98)

Les sept premiers champs de l'exemple d'entrée du format de journal combiné de l'exemple 21-2 sont identiques à ceux du format de journal commun (voir la première entrée de l'exemple 21-1). Les deux nouveaux champs, Référent et Agent utilisateur, sont ajoutés à la fin de l'entrée.

Format de journal étendu Netscape

Lorsque Netscape a fait son entrée sur le marché des applications HTTP commerciales, l'entreprise a défini pour ses serveurs de nombreux formats de journaux, adoptés par d'autres développeurs d'applications HTTP. Les formats Netscape dérivent du format de journal commun NCSA, mais étendent ce format pour y intégrer des champs spécifiques aux applications HTTP, telles que les proxys et les caches web.

Les sept premiers champs du format de journal étendu Netscape sont identiques à ceux du format de journal commun (voir tableau 21-1). Le tableau 21-3 répertorie, par ordre, les nouveaux champs introduits par le format de journal étendu Netscape.

Tableau 21-3. Champs supplémentaires du format de journal étendu Netscape

Description du champ

proxy-response-code Si la transaction est passée par un proxy, le code de réponse HTTP du serveur au proxy

proxy-response-size Si la transaction est passée par un proxy, la longueur du contenu de l'entité de réponse du serveur envoyée au proxy

client-request-size La longueur du contenu de tout corps ou entité dans la demande du client au proxy

proxy-request-size Si la transaction est passée par un proxy, la longueur du contenu de tout corps ou entité dans la requête du proxy au serveur

client-request-hdr-size La longueur, en octets, des en-têtes de requête du client

Tableau 21-3. Champs supplémentaires du format de journal étendu Netscape (suite)

Description du champ

proxy-response-hdr-size Si la transaction est passée par un proxy, la longueur, en octets, des en-têtes de réponse du proxy qui ont été envoyés au demandeur

proxy-request-hdr-size Si la transaction est passée par un proxy, la longueur, en octets, des en-têtes de requête du proxy qui ont été envoyés au serveur

server-response-hdr-size La longueur, en octets, des en-têtes de réponse du serveur

proxy-timestamp Si la transaction est passée par un proxy, le temps écoulé pour que la demande et la réponse transitent par le proxy, en secondes

L'exemple 21-3 donne un exemple d'entrée au format de journal étendu Netscape.

Exemple 21-3. Format de journal étendu Netscape

209.1.32.44 - - [03/oct./1999:14:16:00-0400] "GET / HTTP/1.0" 200 1024 200 1024 0 0 215 260 279 254 3

Dans cet exemple, les champs étendus sont attribués comme suit :

Valeur du champ

code de réponse proxy 200

taille de réponse proxy 1024

taille de la demande client 0

taille de la requête proxy 0

client-request-hdr-size-215

proxy-réponse-hdr-taille-260

proxy-request-hdr-size-279

réponse-serveur-hdr-taille-254

horodatage proxy 3

Les sept premiers champs de l'exemple d'entrée de format de journal étendu Netscape dans l'exemple 21-3 reflètent les entrées de l'exemple de format de journal commun (voir la première entrée dans l'exemple 21-1).

Format de journal Netscape Extended 2

Un autre format de journal Netscape, le Netscape Extended 2 Log Format, reprend le format Extended Log Format et y ajoute des informations complémentaires relatives aux applications de proxy HTTP et de mise en cache web. Ces champs supplémentaires permettent de mieux comprendre les interactions entre un client HTTP et une application proxy HTTP.

Le format de journal étendu Netscape 2 dérive du format de journal étendu Netscape et ses champs initiaux sont identiques à ceux répertoriés dans le tableau 21-3 (il étend également les champs du format de journal commun répertoriés dans le tableau 21-1).

Le tableau 21-4 répertorie, par ordre chronologique, les champs supplémentaires du format de journal Netscape Extended 2. Tableau 21-4. Champs supplémentaires du format de journal Netscape Extended 2.

Description du champ

route La route que le proxy a utilisée pour effectuer la demande pour le client (voir Tableau 21-5)

client-finish-status-code Le code d'état de fin du client ; spécifie si la demande du client au proxy s'est terminée avec succès (FIN) ou a été interrompue (INTR)

proxy-finish-status-code Le code d'état de fin du proxy ; spécifie si la demande de proxy au serveur s'est terminée avec succès (FIN) ou a été interrompue (INTR)

code-résultat-du-cache Le code de résultat du cache ; indique comment le cache a répondu à la requête

a Le tableau 21-7 répertorie les codes de résultat du cache Netscape.

L'exemple 21-4 donne un exemple d'entrée au format Netscape Extended 2 Log.

Exemple 21-4. Format de journal Netscape Extended 2

209.1.32.44 - - [03/Oct/1999:14:16:00-0400] "GET / HTTP/1.0" 200 1024 200 1024 0 0 215 260 279 254 3 DIRECT FIN FIN ÉCRIT

Les champs étendus dans cet exemple sont attribués comme suit :

Valeur du champ

itinéraire DIRECT

code d'état de fin du client FIN

code d'état de fin de proxy FIN

code-résultat-cache ÉCRIT

Les 16 premiers champs de l'entrée Netscape Extended 2 Log Format de l'exemple 21-4 reflètent les entrées de l'exemple Netscape Extended Log Format (voir l'exemple 21-3).

Le tableau 21-5 répertorie les codes de route Netscape valides.

Tableau 21-5. Codes de route Netscape

Description de la valeur

DIRECT La ressource a été récupérée directement depuis le serveur.

PROXY(hôte : port) La ressource a été récupérée via le proxy « hôte ».

SOCKS(socks:port) La ressource a été récupérée via le serveur SOCKS « hôte ».

Le tableau 21-6 répertorie les codes de finition Netscape valides.

Tableau 21-6. Codes d'état de fin de Netscape

Description de la valeur

- La demande n'a même jamais commencé.

FIN La demande a été complétée avec succès.

Tableau 21-6. Codes d'état de fin de Netscape (suite)

Description de la valeur

INTR La demande a été interrompue par le client ou terminée par un proxy/serveur.

TIMEOUT La demande a été expirée par le proxy/serveur.

Le tableau 21-7 répertorie les codes de cache Netscape valides.*

Tableau 21-7. Codes de cache Netscape

Description du code

- La ressource n'était pas cachable.

ÉCRIT La ressource a été écrite dans le cache.

ACTUALISER La ressource a été mise en cache et actualisée.

NO-CHECK La ressource mise en cache a été renvoyée ; aucune vérification de fraîcheur n'a été effectuée.

À JOUR La ressource mise en cache a été renvoyée ; une vérification de fraîcheur a été effectuée.

HÔTE-NON-DISPONIBLE La ressource mise en cache a été renvoyée ; aucune vérification de fraîcheur n'a été effectuée car le serveur distant n'était pas disponible.

CL-MISMATCH La ressource n'a pas été écrite dans le cache ; l'écriture a été abandonnée car la longueur du contenu ne correspondait pas à la taille de la ressource.

ERREUR La ressource n'a pas été écrite dans le cache en raison d'une erreur ; par exemple, un délai d'attente s'est produit ou le client a abandonné la transaction.

Les applications Netscape, comme de nombreuses autres applications HTTP, proposent également d'autres formats de journaux, notamment un format de journal flexible et la possibilité pour les administrateurs de générer des champs de journal personnalisés. Ces formats offrent aux administrateurs un meilleur contrôle et la possibilité de

personnaliser leurs journaux en choisissant les éléments de la transaction HTTP (en-têtes, statut, tailles, etc.) à consigner dans leurs journaux.

La possibilité pour les administrateurs de configurer des formats personnalisés a été ajoutée, car il est difficile de prédire les informations qu'ils souhaiteront obtenir de leurs journaux. De nombreux autres proxys et serveurs permettent également d'émettre des journaux personnalisés.

Format du journal proxy Squid

Le cache proxy Squid (http://www.squid-cache.org) est un élément important du Web. Ses origines remontent à l'un des premiers projets de cache proxy web (ftp://ftp.cs.colorado.edu/pub/techreports/schwartz/Harvest.Conf.ps.Z). Squid est un projet open source qui a été étendu et amélioré par la communauté open source au fil des ans. De nombreux outils ont été développés pour faciliter l'administration de l'application Squid, notamment pour traiter, auditer et exploiter ses journaux. De nombreux caches proxy ultérieurs ont adopté le

Format Squid pour leurs propres journaux afin qu'ils puissent exploiter ces outils.

* Le chapitre 7 décrit en détail la mise en cache HTTP.

Le format d'une entrée de journal Squid est relativement simple. Ses champs sont résumés dans le tableau 21-8.

Tableau 21-8. Champs du format du journal Squid

Description du champ

horodatage L'horodatage d'arrivée de la demande, en secondes depuis le 1er janvier 1970 GMT.

temps écoulé Le temps écoulé pour que la requête et la réponse transitent par le proxy, en millisecondes.

host-ip L'adresse IP de la machine hôte du client (du demandeur).

Le champ de résultat est un Squid-isme qui indique quelle action le proxy a entreprise pendant cette requête ; le champ de code est le code de réponse HTTP que le proxy a envoyé au client.

taille La longueur de la réponse du proxy au client, y compris les en-têtes et le corps de la réponse HTTP, en octets.

méthode La méthode HTTP de la requête du client.

url L'URL dans la demande du client.b

rfc931-identc Le nom d'utilisateur authentifié du client.d

hiérarchie/de Comme le champ route dans les formats Netscape, le champ hiérarchie indique quelle route le proxy a utilisée pour effectuer la demande pour le client. Le champ de indique le nom du serveur que le proxy a utilisé pour effectuer la demande.

type de contenu Le type de contenu de l'entité de réponse proxy.

Le tableau 21-9 répertorie les différents codes de résultats et leur signification.

b Rappelez-vous du chapitre 2 que les proxys enregistrent souvent l'intégralité de l'URL demandée. Par conséquent, si un nom d'utilisateur et un mot de passe sont présents dans l'URL, un proxy peut enregistrer ces informations par inadvertance. c Les champs rfc931-ident, hierarchy/from et content-type ont été ajoutés dans Squid 1.1. Les versions précédentes ne disposaient pas de ces champs. d La RFC 931 décrit la recherche d'identifiant utilisée dans cette authentification. e http://squid.nlanr.net/Doc/FAQ/FAQ-6.html#ss6.6 répertorie tous les codes de hiérarchie Squid valides.

L'exemple 21-5 donne un exemple d'entrée de format Squid Log.

Exemple 21-5. Format du journal Squid

99823414 3001 209.1.32.44 TCP_MISS/200 4087 OBTENIR http://www.joes-hardware.com - DIRECT/ proxy.com texte/html

Les champs sont attribués comme suit :

Valeur du champ

horodatage 99823414

temps écoulé 3001

adresse IP de l'hôte 209.1.32.44

code d'action TCP_MISS

statut 200

taille 4087

méthode GET

URL http://www.joes-hardware.com

Valeur du champ

Identifiant RFC 931 -

hiérarchie DIRECTa

de proxy.com

type de contenu texte/html

a La valeur de la hiérarchie DIRECT Squid est la même que la valeur de la route DIRECT dans les formats de journal Netscape.

Le tableau 21-9 répertorie les différents codes de résultat Squid.*

Tableau 21-9. Codes de résultat Squid

Description de l'action

TCP_HIT Une copie valide de la ressource a été servie à partir du cache.

TCP_MISS La ressource n'était pas dans le cache.

TCP_REFRESH_HIT : la ressource était dans le cache, mais sa fraîcheur devait être vérifiée. Le proxy a revalidé la ressource auprès du serveur et a constaté que la copie en cache était toujours à jour.

TCP_REF_FAIL_HIT : la ressource était dans le cache, mais sa fraîcheur devait être vérifiée. Cependant, la revalidation a échoué (le proxy n'a peut-être pas pu se connecter au serveur), ce qui a renvoyé la ressource « périmée ».

TCP_REFRESH_MISS : la ressource était dans le cache, mais sa fraîcheur devait être vérifiée. Après vérification auprès du serveur, le proxy a constaté que la ressource en cache était obsolète et a reçu une nouvelle version.

TCP_CLIENT_REFRESH_MISS Le demandeur a envoyé une directive Pragma: no-cache ou Cache-Control similaire, de sorte que le proxy a été forcé de récupérer la ressource.

TCP_IMS_HIT Le demandeur a émis une demande conditionnelle, qui a été validée par rapport à la copie mise en cache de la ressource.

TCP_SWAPFAIL_MISS Le proxy pensait que la ressource était dans le cache mais pour une raison quelconque, il n'a pas pu y accéder.

TCP_NEGATIVE_HIT : une réponse en cache a été renvoyée, mais la réponse était négative. Squid prend en charge la mise en cache des erreurs de ressources (par exemple, une réponse 404 introuvable). Ainsi, si plusieurs requêtes transitent par le cache proxy pour une ressource non valide, l'erreur est générée depuis le cache proxy.

TCP_MEM_HIT Une copie valide de la ressource a été servie à partir du cache et la ressource était dans la mémoire du cache proxy (au lieu de devoir accéder au disque pour récupérer la ressource mise en cache).

TCP_DENIED La demande pour cette ressource a été refusée, probablement parce que le demandeur n'a pas l'autorisation de faire des demandes pour cette ressource.

TCP_OFFLINE_HIT : la ressource demandée a été récupérée du cache en mode hors ligne. Les ressources ne sont pas validées lorsque Squid (ou un autre proxy utilisant ce format) est en mode hors ligne.

UDP_* : les codes UDP_* indiquent que les requêtes ont été reçues via l'interface UDP vers le proxy. HTTP utilise normalement le protocole de transport TCP ; ces requêtes n'utilisent donc pas le protocole HTTP.

* Plusieurs de ces codes d'action traitent davantage des aspects internes du cache proxy Squid, donc tous ne sont pas utilisés par d'autres proxys qui implémentent le format de journal Squid.

Tableau 21-9. Codes de résultat Squid (suite)

Description de l'action

UDP_HIT Une copie valide de la ressource a été servie à partir du cache.

UDP_MISS La ressource n'était pas dans le cache.

UDP_DENIED La demande pour cette ressource a été refusée, probablement parce que le demandeur n'a pas l'autorisation de faire des demandes pour cette ressource.

UDP_INVALID La demande reçue par le proxy n'était pas valide.

UDP_MISS_NOFETCH Utilisé par Squid lors de modes de fonctionnement spécifiques ou en cas d'échecs fréquents dans le cache. Un échec de cache a été renvoyé et la ressource n'a pas été récupérée.

AUCUN Enregistré parfois avec des erreurs.

TCP_CLIENT_REFRESH Voir TCP_CLIENT_REFRESH_MISS.

TCP_SWAPFAIL Voir TCP_SWAPFAIL_MISS.

UDP_RELOADING Voir UDP_MISS_NOFETCH.

Un Squid possède son propre protocole pour effectuer ces requêtes : ICP. Ce protocole est utilisé pour les requêtes de cache à cache. Voir http://www.squid-cache.org

pour plus d'informations.

Mesure des hits

Les serveurs d'origine conservent souvent des journaux détaillés à des fins de facturation. Les fournisseurs de contenu doivent connaître la fréquence d'accès aux URL, les annonceurs veulent connaître la fréquence d'affichage de leurs publicités et les auteurs web veulent connaître la popularité de leur contenu. La journalisation est un outil efficace pour suivre ces informations lorsque les clients consultent directement les serveurs web.

Cependant, les caches se situent entre les clients et les serveurs et empêchent de nombreux accès d'atteindre les serveurs (le but même

des caches).* Étant donné que les caches gèrent de nombreuses requêtes HTTP et les satisfont sans visiter le serveur d'origine, le serveur n'a aucun enregistrement qu'un client a accédé à son contenu, créant des omissions dans les fichiers journaux.

L'absence de données de journalisation oblige les fournisseurs de contenu à recourir au contournement du cache pour leurs pages les plus importantes. Ce contournement consiste pour un producteur de contenu à rendre intentionnellement certains contenus non cachables, de sorte que toutes les requêtes pour ce contenu doivent être transmises au serveur d'origine.† Cela permet à ce dernier de consigner les accès. Déjouer la mise en cache peut permettre d'obtenir de meilleurs journaux, mais cela ralentit les requêtes et augmente la charge sur le serveur d'origine et le réseau.

Étant donné que les caches proxy (et certains clients) conservent leurs propres journaux, si les serveurs pouvaient accéder à ces journaux, ou au moins disposer d'un moyen simple de déterminer la fréquence à laquelle leur contenu est servi par un cache proxy, le contournement du cache pourrait être évité. Le protocole Hit Metering, une extension de HTTP, propose une solution à ce problème. Ce protocole exige que les caches transmettent régulièrement les statistiques d'accès au cache aux serveurs d'origine.

* N’oubliez pas que pratiquement tous les navigateurs disposent d’un cache.

† Le chapitre 7 décrit comment les réponses HTTP peuvent être marquées comme non cachables.

La RFC 2227 définit en détail le protocole Hit Metering. Cette section propose un bref aperçu de la proposition.

Aperçu

Le protocole Hit Metering définit une extension de HTTP qui fournit quelques fonctionnalités de base que les caches et les serveurs peuvent implémenter pour partager les informations d'accès et réguler le nombre de fois que les ressources mises en cache peuvent être utilisées.

Le Hit Metering n'est pas, par nature, une solution complète au problème posé par les caches pour la journalisation des accès, mais il fournit un moyen simple d'obtenir les métriques que les serveurs souhaitent suivre. Le protocole Hit Metering n'a pas été largement implémenté ni déployé (et ne le sera peut-être jamais). Cela dit, un système coopératif comme Hit Metering offre la possibilité de fournir des statistiques d'accès précises tout en conservant les gains de performances de la mise en cache. Espérons que cela incitera à implémenter le protocole Hit Metering plutôt que de marquer le contenu comme non cachable.

L'en-tête du compteur

L'extension Hit Metering propose l'ajout d'un nouvel en-tête, Meter, que les caches et les serveurs peuvent utiliser pour se transmettre des directives sur l'utilisation et les rapports, tout comme l'en-tête Cache-Control permet d'échanger des directives de mise en cache.

Le tableau 21-10 définit les différentes directives et les personnes autorisées à les transmettre dans l'en-tête du compteur. Tableau 21-10. Directives de mesure des hits

Directive Abréviation Qui Description

will-report-and-limit w Cache Le cache est capable de signaler l'utilisation et de respecter toutes les limites d'utilisation spécifiées par le serveur.

wont-report x Cache Le cache est capable de respecter les limites d'utilisation mais ne signalera pas l'utilisation.

wont-limit y Cache Le cache est capable de signaler l'utilisation mais ne limitera pas l'utilisation.

count c Cache La directive de reporting, spécifiée comme « utilise/réutilise » des entiers, par exemple « :count=2/4 ».a

max-uses u Serveur Permet au serveur de spécifier le nombre maximal de fois qu'une réponse peut être utilisée par un cache, par exemple, « max-uses=100 ».

max-reuses r Serveur Permet au serveur de spécifier le nombre maximal de fois qu'une réponse peut être réutilisée par un cache, par exemple « max-reuses=100 ».

do-report d Serveur Le serveur nécessite des proxys pour envoyer des rapports d'utilisation.

dont-report e Server Le serveur ne veut pas de rapports d'utilisation.

timeout t Serveur Permet au serveur de spécifier un délai d'expiration pour la mesure d'une ressource. Le cache doit envoyer un rapport avant ou après le délai spécifié, plus ou moins 1 minute. Le délai d'expiration est spécifié en minutes, par exemple « timeout=60 ».

wont-ask n Server Le serveur ne veut aucune information de mesure.

Un Hit Metering définit une utilisation comme la satisfaction d'une demande avec la réponse, tandis qu'une réutilisation consiste à revalider une demande client.

Mesure des hits

La figure 21-1 montre un exemple de mesure des accès en action. La première partie de la transaction est une simple transaction HTTP entre un client et le cache du proxy, mais dans la requête proxy, notez

l'insertion de l'en-tête Meter et de la réponse du serveur. Ici, le proxy informe le serveur qu'il est capable de mesurer les accès, et le serveur demande à son tour au proxy de lui communiquer le nombre d'accès.

Figure 21-1. Exemple de mesure des hits

La requête se termine normalement du point de vue du client, et le proxy commence à suivre les accès à cette ressource pour le compte du serveur. Il tente ensuite de revalider la ressource auprès du serveur. Il intègre les informations mesurées qu'il a suivies dans la requête conditionnelle adressée au serveur.

Un mot sur la confidentialité

La journalisation étant une fonction administrative des serveurs et des proxys, l'opération est entièrement transparente pour les utilisateurs. Souvent, ils ne se rendent même pas compte que leurs transactions HTTP sont journalisées ; en fait, de nombreux utilisateurs ignorent probablement qu'ils utilisent le protocole HTTP lorsqu'ils accèdent à du contenu Web.

Les développeurs et administrateurs d'applications web doivent être conscients des implications du suivi des transactions HTTP d'un utilisateur. Les informations récupérées permettent d'en apprendre beaucoup sur un utilisateur. Ces informations peuvent évidemment être utilisées à mauvais escient : discrimination, harcèlement, chantage, etc. Les serveurs web et les proxys qui enregistrent les données doivent veiller à protéger la confidentialité de leurs utilisateurs finaux.

Parfois, comme dans les environnements de travail, il peut être approprié de suivre l'utilisation d'un utilisateur pour s'assurer qu'il ne fait pas de bêtises, mais les administrateurs doivent également rendre public le fait que les transactions des personnes sont surveillées.

En bref, la journalisation est un outil très utile pour l’administrateur et le développeur. Soyez simplement conscient des atteintes à la vie privée que les journaux peuvent comporter sans l’autorisation ou la connaissance des utilisateurs dont les actions sont enregistrées.

Pour plus d'informations

Pour plus d'informations sur la journalisation, reportez-vous à :

http://httpd.apache.org/docs/logs.html

« Serveur HTTP Apache : fichiers journaux. » Site Web du projet Apache HTTP Server.

http://www.squid-cache.org/Doc/FAQ/FAQ-6.html

« Fichiers journaux Squid ». Site Web de Squid Proxy Cache.

http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format

« Contrôle de la journalisation dans httpd du W3C. »
http://www.w3.org/TR/WD-logfile.html

« Format de fichier journal étendu ».
http://www.ietf.org/rfc/rfc2227.txt

RFC 2227, « Mesure simple des accès et limitation de l'utilisation pour HTTP », par J. Mogul et P. Leach.

Pour plus d'informations

PARTIE VI

VI. Annexes

Cette collection d'annexes contient des tableaux de référence utiles, des informations générales et des didacticiels sur une variété de sujets relatifs à l'architecture et à la mise en œuvre HTTP :

• Annexe A, Schémas URI

• Annexe B, codes d'état HTTP

• Annexe C, Référence d'en-tête HTTP

• Annexe D, Types MIME

• Annexe E, Codage en base 64

• Annexe F, Authentification Digest

• Annexe G, Balises de langue

• Annexe H, Registre des jeux de caractères MIME

Annexe ACeci est le titre du livre ANNEXE A

Schémas URI

De nombreux schémas d'URI ont été définis, mais peu sont couramment utilisés. En général, les schémas d'URI et leurs RFC associés sont les plus courants, bien que quelques schémas, développés par des sociétés de logiciels de premier plan (notamment

Netscape et Microsoft), mais non formalisés, soient également largement utilisés.

Le W3C maintient une liste de schémas URI, que vous pouvez consulter à l'adresse :

http://www.w3.org/Addressing/schemes.html

L'IANA maintient également une liste de schémas d'URL, à l'adresse :

http://www.iana.org/assignments/uri-schemes

Le tableau A-1 décrit de manière informelle certains des schémas proposés et ceux actuellement utilisés. Il convient de noter que parmi les quelque 90 schémas présentés, beaucoup ne sont pas largement utilisés et que beaucoup ont disparu.

Tableau A-1. Schémas d'URI du registre W3C

Description du schéma RFC

À propos du schéma Netscape pour explorer les différents aspects du navigateur. Par exemple : « À propos » équivaut à choisir « À propos de Communicator » dans le menu Aide de Navigator, « À propos : cache » affiche les statistiques du cache disque et « À propos : plugins » affiche des informations sur les plugins configurés. D'autres navigateurs, comme Microsoft Internet Explorer, utilisent également ce schéma.

Protocole d'accès à la configuration de l'application acap. 2244

afp Pour les services de partage de fichiers utilisant le protocole Apple Filing Protocol (AFP), défini dans le cadre du projet IETF expiré-ietf-svrloc-afp-service-01.txt.

afs Réservé pour une utilisation future par le système de fichiers Andrew.

callto Lance une session de conférence Microsoft NetMeeting, telle que :

appel à : ws3.joes-hardware.com/ joe@joes-hardware.com

chttp : protocole de mise en cache CHTTP défini par Real Networks. RealPlayer ne met pas en cache tous les éléments diffusés via HTTP. Il désigne les fichiers à mettre en cache en utilisant chttp:// au lieu de http:// dans l'URL du fichier. Lorsque RealPlayer lit une URL CHTTP dans un fichier SMIL, il vérifie d'abord le fichier dans son cache disque. Si le fichier est absent, il le demande via HTTP et le stocke dans son cache.

Description du schéma RFC

L'utilisation de [MIME] dans les e-mails pour transmettre des pages web et leurs images nécessite un schéma d'URL permettant au code

HTML de faire référence aux images ou autres données incluses dans le message. L'URL Content-ID, « cid: », remplit cette fonction. 2392

2111

clsid permet de référencer les classes Microsoft OLE/COM (Component Object Model). Permet d'insérer des objets actifs dans des pages web.

Données : permet l'inclusion de petits éléments de données constants comme données « immédiates ». Cette URL encode le texte/chaîne simple « Note brève » : données : Note brève 2397

Proposition de schéma pour prendre en charge les dates, comme dans date:1999-03-04T20:42:08.

dav Pour garantir une interopérabilité correcte basée sur cette spécification, l'IANA doit réserver les espaces de noms URI commençant par « DAV : » et par « opaquelocktoken : » pour une utilisation par cette spécification, ses révisions et les spécifications WebDAV associées. 2518

DNS Utilisé par le logiciel REBOL.

Voir http://www.rebol.com/users/valurl.html.

Le schéma d'identifiant externe (eid) fournit un mécanisme permettant à l'application locale de référencer des données obtenues par d'autres moyens, autres que des URL. Ce schéma vise à fournir un mécanisme d'échappement général permettant l'accès aux informations pour les applications trop spécialisées pour justifier leurs propres schémas. Cet URI fait l'objet d'une controverse.

Voir http://www.ics.uci.edu/pub/ietf/uri/draft-finseth-url-00.txt.

Le schéma « fax » décrit une connexion à un terminal capable de gérer les télécopies (télécopieurs).

Fichier : désigne les fichiers accessibles sur un ordinateur hôte particulier. Un nom d'hôte peut être inclus, mais ce schéma est inhabituel car il ne spécifie pas de protocole Internet ni de méthode d'accès pour ces fichiers ; son utilité dans les protocoles réseau entre hôtes est donc limitée. 1738

L'URL du doigt a la forme :

doigt://hôte[:port][/<request>]

La <request> doit être conforme au format de demande RFC 1288.

Voir http://www.ics.uci.edu/pub/ietf/uri/draft-ietf-uri-url-finger-03.txt.

URI Freenet pour les informations dans le système d'information distribué Freenet.

Voir http://freenet.sourceforge.net.

Schéma du protocole de transfert de fichiers ftp. 1738

gopher Le protocole archaïque du gopher. 1738

URI gsm-sms pour le service de messages courts du téléphone mobile GSM.

Schémas URI de conférence multimédia h323, h324.

Voir http://www.ics.uci.edu/pub/ietf/uri/draft-cordell-sg16-conv-url-00.txt.

Le système Handle est un système complet d'attribution, de gestion et de résolution d'identifiants persistants, appelés « handles », pour les objets numériques et autres ressources sur Internet.

Les poignées peuvent être utilisées comme URN.

Voir http://www.handle.net.

hnews est une variante HTTP tunnel du protocole d'actualités NNTP. La syntaxe des URL hnews est conçue pour être compatible avec l'usage courant du schéma d'URL d'actualités.

Voir http://www.ics.uci.edu/pub/ietf/uri/draft-stockwell-hnews-url-00.txt.

Description du schéma RFC

http Le protocole HTTP. Pour en savoir plus, consultez ce livre.

https HTTP sur SSL.

Voir http://sitesearch.netscape.com/eng/ssl3/draft302.txt.

Extensions CORBA iioploc. Le service de noms interopérables définit une référence d'objet au format URL, iioploc, qui peut être saisie dans un programme pour accéder à des services définis à distance, y compris le service de noms. Par exemple, cet identifiant iioploc :

iioploc://www.omg.org/NameService

résoudrait le service de nommage CORBA exécuté sur la machine dont l'adresse IP correspond au nom de domaine www.omg.org.

Voir http://www.omg.org.

Le système d'unification interlinguistique (ILU) est un système d'interface objet multilingue. Les interfaces objet fournies par ILU masquent les différences d'implémentation entre les différents langages, espaces d'adressage et types de systèmes d'exploitation. ILU permet de créer des bibliothèques orientées objet multilingues (« bibliothèques de classes ») dotées d'interfaces bien spécifiées et indépendantes du langage. Il permet également d'implémenter des systèmes distribués.

Voir ftp://parcftp.parc.xerox.com/pub/ilu/ilu.html.

Le schéma d'URL IMAP est utilisé pour désigner les serveurs IMAP, les boîtes aux lettres, les messages, les corps MIME [MIME] et les programmes de recherche sur les hôtes Internet accessibles à l'aide du protocole IMAP.

Référence d'objet interopérable IOR CORBA.

Voir http://www.omg.org.

Le schéma d'URL irc est utilisé pour faire référence soit aux serveurs Internet Relay Chat (IRC), soit aux entités individuelles (canaux ou personnes) sur les serveurs IRC.

Voir http://www.w3.org/Addressing/draft-mirashi-url-irc-01.txt.

isbn Schéma proposé pour les références de livres ISBN.

Voir http://lists.w3.org/Archives/Public/www-talk/1991NovDec/0008.html.

java Identifie les classes Java.

Le navigateur Netscape traite les URL javascript, évalue l'expression après les deux points (:), s'il y en a une, et charge une page contenant la valeur de chaîne de l'expression, sauf si elle n'est pas définie.

jdbc Utilisé dans l'API Java SQL.

ldap Permet aux clients Internet d'accéder directement au protocole LDAP. 2255

couvercle Le schéma d'identifiant local (lid :).

Voir draft-blackketter-lid-00.

lifn Un nom de fichier indépendant de l'emplacement (LIFN) pour le système de stockage distribué Bulk File Distribution développé à l'UTK.

livescript Ancien nom de JavaScript.

lrq Voir h323.

Le schéma d'URL mailto est utilisé pour désigner l'adresse de messagerie Internet d'un individu ou d'un service.

serveur de messagerie Ancienne proposition de 1994-1995 permettant d'encoder un message entier dans une URL, afin que (par exemple) l'URL puisse envoyer automatiquement un courrier électronique à un serveur de messagerie pour s'abonner à une liste de diffusion.

Schémas URI |

Description du schéma RFC

md5 MD5 est une somme de contrôle cryptographique.

Le schéma mid utilise (une partie de) l'ID de message d'un message électronique pour faire référence à un message spécifique.

2111

moka Voir javascript.

Le schéma du modem décrit une connexion à un terminal capable de gérer les appels de données entrants.

Schéma mms, mmst, mmsu pour Microsoft Media Server (MMS) permettant de diffuser des fichiers ASF (Active Streaming Format). Pour forcer le transport UDP, utilisez le schéma mmsu. Pour forcer le transport TCP, utilisez mmst.

L'URL d'actualités est utilisée pour désigner des groupes de discussion ou des articles d'actualités USENET. Une URL d'actualités prend deux formes : news:<newsgroup-name> ou news:<message-id>. 1738

1036

nfs Utilisé pour faire référence aux fichiers et répertoires sur les serveurs NFS.

nntp : une méthode alternative de référencement d'articles d'actualité, utile pour spécifier les articles d'actualité provenant de serveurs NNTP. Une URL nntp ressemble à ceci :

nntp://<hôte>:<port>/<nom-du-groupe-de-presse>/<numéro-d'article>

Notez que, bien que les URL NNTP spécifient un emplacement unique pour la ressource article, la plupart des serveurs NNTP actuellement sur Internet sont configurés pour autoriser l'accès uniquement depuis les clients locaux ; par conséquent, les URL NNTP ne désignent pas les ressources accessibles globalement. Par conséquent, la forme d'URL « news » est privilégiée pour identifier les articles d'actualité.

977

opaquelocktoken : jeton de verrouillage WebDAV, représenté par un URI, qui identifie un verrou particulier. Un jeton de verrouillage est renvoyé par chaque opération LOCK réussie dans la propriété lockdiscovery du corps de la réponse et peut également être trouvé via la découverte de verrous sur une ressource. Voir RFC 2518.

Le schéma de chemin définit un espace de noms uniformément hiérarchique où un URN de chemin est une séquence de composants et une chaîne opaque facultative.

Voir http://www.hypernews.org/~liberte/www/path.html.

téléphone Utilisé dans « URL pour la téléphonie » ; remplacé par tel: dans la RFC 2806.

L'URL POP désigne un serveur de messagerie POP et éventuellement un numéro de port, un mécanisme d'authentification, un ID d'authentification et/ou un ID d'autorisation. 2384

Protocole de streaming de pnm Real Networks.

pop3 Le schéma d'URL POP3 permet à une URL de spécifier un serveur POP3, permettant ainsi à d'autres protocoles d'utiliser une « URL à utiliser pour l'accès au courrier » générale au lieu d'une référence explicite à POP3. Défini dans le fichier draft-earhart-url-pop3-00.txt expiré.

URL abstraites de l'imprimante à utiliser avec la norme Service Location.

Voir draft-ietf-srvloc-printer-scheme-02.txt.

prospero Nomme les ressources accessibles via le service d'annuaire Prospero. 1738

Schéma Microsoft res spécifiant une ressource à obtenir à partir d'un module. Il se compose d'un type de ressource chaîne ou numérique et d'un identifiant chaîne ou numérique.

Protocole de streaming en temps réel qui constitue la base des protocoles de contrôle de streaming modernes de Real Networks.

URL rvp pour le protocole de rendez-vous RVP, utilisé pour notifier l'arrivée des utilisateurs sur un réseau informatique.

Voir draft-calsyn-rvp-01.

Description du schéma RFC

rwhois RWhois est un protocole d'accès aux annuaires Internet, défini dans les RFC 1714 et RFC 2167. Le RWhois

L'URL donne aux clients un accès direct à rwhois.

Voir http://www.rwhois.net/rwhois/docs/.

rx Une architecture permettant aux applications graphiques distantes d'afficher des données dans des pages Web.

Voir http://www.w3.org/People/danield/papers/mobgui/.

URL du protocole de description de session (SDP) sdp. Voir RFC 2327.

Service : le schéma de service permet de fournir des informations d'accès à des services réseau arbitraires. Ces URL offrent un cadre extensible permettant aux logiciels réseau clients d'obtenir les informations de configuration nécessaires à l'utilisation des services réseau.

La famille de schémas sip* est utilisée pour établir des conférences multimédias à l'aide du protocole d'initiation de session (SIP).

shttp (S-HTTP) est un sur-ensemble de HTTP conçu pour sécuriser les connexions HTTP et fournir une grande variété de mécanismes garantissant la confidentialité, l'authentification et l'intégrité. Peu déployé, il a été largement supplanté par HTTPS (HTTP chiffré par SSL).

Voir http://www.homeport.org/~adam/shttp.html.

snews Actualités cryptées SSL.

STANF : ancienne proposition de noms de fichiers réseau stables. Liée aux URN.

Voir http://web3.w3.org/Addressing/#STANF.

t120 Voir h323.

tel URL pour passer un appel en utilisant le réseau téléphonique. 2806

téléphone Utilisé dans les versions précédentes de tél.

Telnet désigne les services interactifs accessibles via le protocole Telnet. Une URL Telnet se présente sous la forme suivante :

telnet://<utilisateur>:<mot de passe>@<hôte>:<port>/ 1738

tip Prend en charge les transactions Internet atomiques TIP.

2372

tn3270 Réservé, selon ftp://ftp.isi.edu/in-notes/iana/assignments/url-schemes.

L'URL TV nomme une chaîne de diffusion télévisée particulière.

Les identifiants uniques universels (UUID) ne contiennent aucune information de localisation. Ils sont également appelés identifiants uniques globaux (GUID). Ils sont persistants dans le temps, comme les URN, et consistent en un identifiant unique de 128 bits. Les URI UUID sont utiles lorsqu'un identifiant unique est requis, mais qu'il ne peut ou ne doit pas être lié à un espace de noms racine physique particulier (comme un nom DNS).

Voir draft-kindel-uuid-uri-00.txt.

urne URN persistantes, indépendantes de l'emplacement. 2141

vemmi permet aux logiciels clients et terminaux VEMMI (Interface multimédia polyvalente) de se connecter à des services compatibles VEMMI. VEMMI est une norme internationale pour les services multimédias en ligne.

vidéotex Permet aux logiciels clients ou terminaux vidéotex de se connecter à des services vidéotex conformes aux normes vidéotex ITU-T et ETSI.

Voir http://www.ics.uci.edu/pub/ietf/uri/draft-mavrakis-videotex-url-spec-01.txt.

Schémas URI |

Description du schéma RFC

Visualiseurs de sources Netscape Navigator. Ces URL de visualisation affichent du code HTML généré avec JavaScript.

wais Le service d'information sur une zone étendue — une forme précoce de moteur de recherche. 1738

URL whois++ pour le protocole d'annuaire Internet simple WHOIS++.

Voir http://martinh.net/wip/whois-url.txt. 1835

Le protocole WhoDP (Widely Hosted Object Data Protocol) permet de communiquer la localisation et l'état actuels d'un grand nombre d'objets dynamiques et relocalisables. Un programme WhoDP s'abonne pour localiser et recevoir des informations sur un objet, et publie des informations pour contrôler sa localisation et son état visible.

Voir draft-mohr-whodp-00.txt.

z39.50r, z39.50s URL de session et de récupération Z39.50. Z39.50 est un protocole de récupération d'informations qui ne s'intègre pas parfaitement dans un modèle de récupération principalement axé sur la récupération de données sans état. Il modélise plutôt une requête utilisateur générale comme une tâche orientée session, en plusieurs étapes, dont chaque étape peut être suspendue temporairement pendant que le serveur demande des paramètres supplémentaires au client avant de poursuivre. 2056

Annexe BCeci est le titre du livre ANNEXE B

Codes d'état HTTP

Cette annexe est une référence rapide des codes d’état HTTP et de leurs significations.

Classifications des codes d'état

Les codes d’état HTTP sont segmentés en cinq classes, présentées dans le tableau B-1.

Tableau B-1. Classifications des codes d'état

Gamme globale Gamme définie Catégorie

100–199 100–101 Informatif

200–299 200–206 Réussi

300–399 300–305 Redirection

400–499 400–415 Erreur client

500–599 500–505 Erreur du serveur

Codes d'état

Le tableau B-2 est un guide de référence rapide pour tous les codes d'état définis dans la spécification HTTP/1.1, fournissant un bref résumé de chacun. La section « Codes d'état » du chapitre 3 décrit plus en détail ces codes d'état et leurs utilisations.

Tableau B-2. Codes d'état

Code d'état Phrase de raison Signification

100 Continuer Une première partie de la demande a été reçue et le client doit continuer.

101 Protocoles de commutation Le serveur change de protocole, comme spécifié par le client, vers un protocole répertorié dans l'en-tête de mise à niveau.

200 OK La demande est correcte.

201 Créé La ressource a été créée (pour les requêtes qui créent des objets serveur).

505

Tableau B-2. Codes d'état (suite)

Code d'état Phrase de raison Signification

202 Accepté La demande a été acceptée, mais le serveur n'a pas encore effectué d'action avec elle.

203 Informations non autorisées La transaction s'est déroulée correctement, sauf que les informations contenues dans les en-têtes d'entité ne provenaient pas du serveur d'origine, mais d'une copie de la ressource.

204 Aucun contenu Le message de réponse contient des en-têtes et une ligne d'état, mais aucun corps d'entité.

205 Réinitialiser le contenu Un autre code principalement destiné aux navigateurs ; cela signifie essentiellement que le navigateur doit effacer tous les éléments de formulaire HTML sur la page actuelle.

206 Contenu partiel Une demande partielle a réussi.

300 Choix multiples : un client a demandé une URL qui renvoie vers plusieurs ressources. Ce code est renvoyé avec une liste d'options ; l'utilisateur peut alors sélectionner celle qu'il souhaite.

301 Déplacé définitivement : l'URL demandée a été déplacée. La réponse doit contenir une URL d'emplacement indiquant où se trouve désormais la ressource.

302 Trouvé : similaire au code d'état 301, mais le déplacement est temporaire. Le client doit utiliser l'URL indiquée dans l'en-tête Emplacement pour localiser temporairement la ressource.

303 Voir Autre Indique au client que la ressource doit être récupérée à l'aide d'un autre

URL. Cette nouvelle URL se trouve dans l'en-tête Emplacement du message de réponse.

304 Non modifié : les clients peuvent conditionner leurs requêtes en fonction des en-têtes de requête inclus. Ce code indique que la ressource n'a pas été modifiée.

305 Utiliser un proxy La ressource doit être accessible via un proxy, l'emplacement du proxy est indiqué dans l'en-tête Location.

306 (inutilisé) Ce code d'état n'est actuellement pas utilisé.

307 Redirection temporaire Comme le code d'état 301 ; cependant, le client doit utiliser l'URL indiquée dans l'en-tête Emplacement pour localiser temporairement la ressource.

400 Bad Request indique au client qu'il a envoyé une demande mal formée.

401 Non autorisé Renvoyé avec les en-têtes appropriés qui demandent au client de s'authentifier avant de pouvoir accéder à la ressource.

402 Paiement requis Actuellement, ce code de statut n'est pas utilisé, mais il a été mis de côté pour une utilisation future.

403 Interdit La demande a été refusée par le serveur.

404 Non trouvé Le serveur ne trouve pas l'URL demandée.

405 Méthode non autorisée : une requête a été effectuée avec une méthode non prise en charge pour l'URL demandée. L'en-tête « Allow » doit être inclus dans la réponse pour indiquer au client les méthodes autorisées sur la ressource demandée.

406 Non acceptable : les clients peuvent spécifier des paramètres concernant les types d'entités qu'ils acceptent. Ce code est utilisé lorsque le serveur ne dispose d'aucune ressource correspondant à l'URL acceptable pour le client.

407 Authentification proxy requise Comme le code d'état 401, mais utilisé pour les serveurs proxy qui nécessitent une authentification pour une ressource.

Annexe B : Codes d'état HTTP

Tableau B-2. Codes d'état (suite)

Code d'état Phrase de raison Signification

408 Délai d'expiration de la demande Si un client prend trop de temps pour terminer sa demande, un serveur peut renvoyer ce code d'état et fermer la connexion.

409 Conflit La requête provoque un conflit sur une ressource.

410 Gone Comme le code d'état 404, sauf que le serveur détenait autrefois la ressource.

411 Longueur requise : les serveurs utilisent ce code lorsqu'ils exigent un en-tête Content-Length dans le message de requête. Le serveur n'acceptera pas les requêtes pour la ressource sans cet en-tête.

412 Échec de la précondition Si un client effectue une demande conditionnelle et que l'une des conditions échoue, ce code de réponse est renvoyé.

413 Entité de demande trop grande Le client a envoyé un corps d'entité plus grand que ce que le serveur peut ou veut traiter.

414 URI de demande trop long Le client a envoyé une demande avec une URL de demande plus grande que ce que le serveur peut ou veut traiter.

415 Type de média non pris en charge Le client a envoyé une entité d'un type de contenu que le serveur ne comprend pas ou ne prend pas en charge.

416 Plage demandée non satisfaisante Le message de demande demandait une plage d'une ressource donnée, et cette plage n'était pas valide ou ne pouvait pas être respectée.

417 Échec de l'attente La demande contenait une attente dans l'en-tête de demande Expect qui n'a pas pu être satisfaite par le serveur.

500 Erreur interne du serveur Le serveur a rencontré une erreur qui l'a empêché de traiter la demande.

501 Non implémenté Le client a effectué une demande qui dépasse les capacités du serveur.

502 Bad Gateway Un serveur agissant comme proxy ou passerelle a rencontré une réponse erronée du lien suivant dans la chaîne de réponse de la demande.

503 Service indisponible Le serveur ne peut actuellement pas traiter la demande, mais pourra le faire à l'avenir.

504 Délai d'expiration de la passerelle Similaire au code d'état 408, sauf que la réponse provient d'une passerelle ou d'un proxy qui a expiré en attendant une réponse à sa demande d'un autre serveur.

505 Version HTTP non prise en charge Le serveur a reçu une demande dans une version du protocole qu'il ne peut pas ou ne veut pas prendre en charge.

Codes d'état

Annexe CCeci est le titre du livreANNEXE C

Référence d'en-tête HTTP

C'est presque amusant de se rappeler que la première version de HTTP, 0.9, n'avait pas d'en-têtes.

Même si cela a certainement ses inconvénients, il est amusant de s’émerveiller de son élégance simpliste.

Bon, revenons à la réalité. Il existe aujourd'hui une multitude d'en-têtes HTTP, dont beaucoup font partie intégrante de la spécification, tandis que d'autres en sont des extensions. Cette annexe fournit quelques informations générales sur ces en-têtes officiels et d'extension. Elle sert également d'index des différents en-têtes de ce livre, indiquant où leurs concepts et fonctionnalités sont abordés dans le texte. La plupart de ces en-têtes sont simples au départ ; c'est dans leurs interactions entre eux et avec les autres fonctionnalités de HTTP que les choses se compliquent. Cette annexe fournit quelques informations générales sur les en-têtes répertoriés et vous dirige vers les sections du livre où ils sont abordés en détail.

Les en-têtes répertoriés dans cette annexe sont tirés des spécifications HTTP, des documents associés et de notre propre expérience en matière de recherche de messages HTTP et des différents serveurs et clients sur Internet.

Cette liste est loin d'être exhaustive. De nombreux autres en-têtes d'extension circulent sur le Web, sans parler de ceux potentiellement utilisés dans les intranets privés. Néanmoins, nous avons tenté de rendre cette liste aussi complète que possible. Consultez la RFC 2616 pour la version actuelle de la spécification HTTP/1.1 et la liste des en-têtes officiels et leurs descriptions.

Accepter

L'en-tête Accept est utilisé par les clients pour indiquer aux serveurs les types de médias acceptés. La valeur du champ d'en-tête Accept est une liste des types de médias que le client peut utiliser. Par exemple, votre navigateur web ne peut pas afficher tous les types d'objets multimédias disponibles sur le Web. En incluant un en-tête Accept dans vos requêtes, votre navigateur peut vous éviter de télécharger une vidéo ou un autre type d'objet inutilisable.

Le champ d'en-tête Accept peut également inclure une liste de valeurs de qualité (valeurs q) indiquant au serveur le type de média préféré, s'il en possède plusieurs versions. Consultez le chapitre 17 pour une analyse complète de la négociation de contenu et des valeurs q.

Type d'en-tête de demande

Remarques : « * » est une valeur spéciale utilisée pour les types de médias génériques. Par exemple :

« */* » représente tous les types et « image/* » représente tous les types d'images.

Exemples Accepter : texte/*, image/*

Accepter : texte/*, image/gif, image/jpeg ; q=1

Accepter-Charset

L'en-tête Accept-Charset est utilisé par les clients pour indiquer aux serveurs les jeux de caractères acceptables ou préférés. La valeur de cet en-tête de requête est une liste de jeux de caractères et, éventuellement, des valeurs de qualité pour ces jeux. Ces valeurs permettent au serveur de savoir quel jeu de caractères est préféré, si le document contient plusieurs jeux de caractères acceptables. Consultez le chapitre 17 pour une discussion complète sur la négociation de contenu et les valeurs q.

Type d'en-tête de demande

Remarques : comme pour l'en-tête Accept, « * » est un caractère spécial. S'il est présent, il représente tous les jeux de caractères, à l'exception de ceux mentionnés explicitement dans la valeur. S'il est absent, tout jeu de caractères non présent dans le champ de valeur a une valeur q par défaut de zéro, à l'exception du jeu de caractères iso-latin-1, qui a une valeur par défaut de 1.

Syntaxe de base Accept-Charset : 1# ((charset | "*") [";" "q" "=" qvalue])

Exemple Accept-Charset : iso-latin-1

Accepter-Encodage

L'en-tête Accept-Encoding est utilisé par les clients pour indiquer aux serveurs les encodages acceptés. Si le contenu stocké par le serveur est encodé (éventuellement compressé), cet en-tête de requête indique au serveur si le client l'accepte. Le chapitre 17 contient une description complète de l'en-tête Accept-Encoding.

Type d'en-tête de demande

Syntaxe de base Accept-Encoding : 1# ((content-coding | "*") [";" "q" "=" qvalue])

Exemples* Accept-Encoding :

Accepter-Encodage : gzip

Accepter-Encodage : compresser ; q = 0,5, gzip ; q = 1

* L'exemple d'en-tête Accept-Encoding vide n'est pas une erreur typographique. Il fait référence à l'encodage d'identité, c'est-à-dire au contenu non encodé. Si l'en-tête Accept-Encoding est présent et vide, seul le contenu non encodé est acceptable.

Accepter-Encodage

Accepter-Langue

L'en-tête de requête Accept-Language fonctionne comme les autres en-têtes Accept, permettant aux clients d'indiquer au serveur les langues (par exemple, la langue naturelle du contenu) acceptées ou préférées. Le chapitre 17 contient une description complète de l'en-tête AcceptLanguage.

Type d'en-tête de demande

Syntaxe de base Accept-Language : 1# (plage-langue [";" "q" "=" qvalue]) plage-langue = ((1*8ALPHA * ("-" 1*8ALPHA)) | "*")

Exemples Accept-Language: en

Accepter la langue : en;q=0.7, en-gb;q=0.5

Plages d'acceptation

L'en-tête Accept-Ranges diffère des autres en-têtes Accept : il s'agit d'un en-tête de réponse utilisé par les serveurs pour indiquer aux clients s'ils acceptent les requêtes portant sur des plages d'une ressource. La valeur de cet en-tête indique le type de plages, le cas échéant, que le serveur accepte pour une ressource donnée.

Un client peut tenter d'effectuer une requête de plage sur une ressource sans avoir reçu cet en-tête. Si le serveur ne prend pas en charge les requêtes de plage pour cette ressource, il peut répondre

avec un code d'état approprié* et la valeur « none » dans Accept-Ranges. Les serveurs peuvent souhaiter envoyer la valeur « none » pour les requêtes normales afin de dissuader les clients d'effectuer des requêtes de plage à l'avenir.

Le chapitre 17 contient une description complète de l'en-tête Accept-Ranges.

Type d'en-tête de réponse

Syntaxe de base Accept-Ranges : 1# range-unit | none

Exemples Accept-Ranges : aucun Accept-Ranges : octets

Âge

L'en-tête « Âge » indique au destinataire l'ancienneté d'une réponse. Il s'agit de la meilleure estimation de l'expéditeur quant à la date à laquelle la réponse a été générée ou revalidée par le serveur d'origine. La valeur de l'en-tête correspond à l'estimation de l'expéditeur, exprimée en secondes. Pour plus d'informations sur l'en-tête « Âge », voir le chapitre 7.

Type d'en-tête de réponse

Les caches HTTP/1.1 doivent inclure un en-tête Age dans chaque réponse qu'ils envoient.

* Par exemple, le code d'état 416 (voir « 400–499 : Codes d'état d'erreur client » au chapitre 3).

Syntaxe de base Âge : delta-secondes

Exemple Âge : 60

Permettre

L'en-tête Allow est utilisé pour informer les clients des méthodes HTTP prises en charge sur une ressource particulière.

Type d'en-tête de réponse

Remarques Un serveur HTTP/1.1 envoyant une réponse 405 Méthode non autorisée doit

inclure un en-tête Allow.*

Syntaxe de base autorisée : #Méthode

Exemple Autoriser : GET, HEAD

Autorisation

L'en-tête d'autorisation est envoyé par un client pour s'authentifier auprès d'un serveur. Un client inclura cet en-tête dans sa requête après avoir reçu une réponse 401 « Authentification requise » d'un serveur. La valeur de cet en-tête dépend du schéma d'authentification utilisé. Voir le chapitre 14 pour une description détaillée de l'en-tête d'autorisation.

Type d'en-tête de réponse

Autorisation de syntaxe de base : authentication-scheme #authentication-param

Exemple d'autorisation : Basic YnJpYW4tdG90dHk6T3ch

Contrôle du cache

L'en-tête Cache-Control permet de transmettre des informations sur la mise en cache d'un objet. Cet en-tête est l'un des plus complexes introduits dans HTTP/1.1. Sa valeur est une directive de mise en cache, donnant aux caches des instructions spécifiques sur la mise en cache d'un objet.

Dans le chapitre 7, nous discutons de la mise en cache en général ainsi que des détails spécifiques concernant cet en-tête.

Type En-tête général

Exemple de contrôle de cache : no-cache

* Voir « Codes d'état » au chapitre 3 pour plus d'informations sur le code d'état 405.

Contrôle du cache

Client-ip

L'en-tête Client-ip est un en-tête d'extension utilisé par certains clients plus anciens et certains proxys pour transmettre l'adresse IP de la machine sur laquelle le client s'exécute.

En-tête de demande d'extension de type

Les implémenteurs doivent être conscients que les informations fournies dans la valeur de cet en-tête ne sont pas sécurisées.

Syntaxe de base Client-ip : adresse IP

Exemple d'adresse IP du client : 209.1.33.49

Connexion

L'en-tête de connexion est un en-tête quelque peu surchargé, ce qui peut prêter à confusion. Cet en-tête était utilisé dans les clients HTTP/1.0, qui intégraient des connexions persistantes pour les informations de contrôle.* Dans HTTP/1.1, l'ancienne sémantique est généralement reconnue, mais l'en-tête a acquis une nouvelle fonction.

En HTTP/1.1, la valeur de l'en-tête de connexion est une liste de jetons correspondant à leurs noms. Les applications recevant un message HTTP/1.1 avec un en-tête de connexion sont censées analyser cette liste et supprimer tous les en-têtes du message qui y figurent. Ceci est principalement destiné aux proxys, permettant à un serveur ou à un autre proxy de spécifier des en-têtes saut par saut à ne pas transmettre.

Une valeur de jeton spéciale est « close ». Ce jeton signifie que la connexion sera fermée une fois la réponse terminée. Les applications HTTP/1.1 qui ne prennent pas en charge les connexions persistantes doivent insérer l'en-tête Connection avec le jeton « close » dans toutes les requêtes et réponses.

Type En-tête général

Remarques Bien que la RFC 2616 ne mentionne pas spécifiquement le keep-alive comme jeton de connexion, certains navigateurs (y compris ceux qui envoient HTTP/1.1 comme versions) l'utilisent pour faire des requêtes.

Syntaxe de base Connexion : 1# (jeton de connexion)

Exemples de connexion : fermer

* Voir le chapitre 4 pour en savoir plus sur les connexions persistantes et persistantes.

Base de contenu

L'en-tête Content-Base permet à un serveur de spécifier une URL de base pour résoudre les URL trouvées dans le corps de l'entité d'une réponse.* La valeur de l'en-tête Content-Base est une URL absolue qui peut être utilisée pour résoudre les URL relatives trouvées à l'intérieur de l'entité.

En-tête d'entité de type

Remarques Cet en-tête n'est pas défini dans la RFC 2616 ; il était auparavant défini dans la RFC 2068, une version antérieure de la spécification HTTP/1.1, et a depuis été supprimé de la spécification officielle.

Syntaxe de base Content-Base : absoluteURL

Exemple de base de contenu : http://www.joes-hardware.com/

Codage de contenu

L'en-tête Content-Encoding permet de spécifier si des encodages ont été effectués sur l'objet. En encodant le contenu, le serveur peut le compresser avant d'envoyer la réponse. La valeur de l'en-tête Content-Encoding indique au client le ou les types d'encodage appliqués à l'objet. Grâce à ces informations, le client peut ensuite décoder le message.

Parfois, plusieurs codages sont appliqués à une entité, auquel cas les codages doivent être répertoriés dans l'ordre dans lequel ils ont été effectués.

En-tête d'entité de type

Syntaxe de base Content-Encoding : 1# content-coding

Exemples de codage de contenu : gzip

Codage de contenu : compresser, gzip

Contenu-Langue

L'en-tête Content-Language indique au client la langue naturelle à comprendre pour comprendre l'objet. Par exemple, un document rédigé en français aurait une valeur Content-Language indiquant le français. Si cet en-tête est absent de la réponse, l'objet est destiné à tous les publics. La présence de plusieurs langues dans la valeur de l'en-tête indique que l'objet est adapté aux publics de chaque langue indiquée.

Un avertissement concernant cet en-tête est que la valeur de l'en-tête peut simplement représenter la langue naturelle du public visé par cet objet, et non toutes ou aucune des langues contenues.

* Voir le chapitre 2 pour plus d’informations sur les URL de base.

Contenu-Langue

dans l'objet. De plus, cet en-tête ne se limite pas aux objets texte ou de données écrites ; les images, vidéos et autres types de médias peuvent être étiquetés avec le langage naturel de leur public cible.

En-tête d'entité de type

Contenu de syntaxe de base - Langue : 1# langue-balise

Exemples Contenu-Langue : en

Contenu-Langue : en, fr

Longueur du contenu

L'en-tête Content-Length indique la longueur ou la taille du corps de l'entité. Si l'en-tête se trouve dans un message de réponse à une requête HTTP HEAD, sa valeur indique la taille qu'aurait eue le corps de l'entité s'il avait été envoyé.

En-tête d'entité de type

Contenu de la syntaxe de base - Longueur : 1*CHIFFRE

Exemple de contenu - Longueur : 2417

Contenu-Emplacement

L'en-tête Content-Location est inclus dans un message HTTP pour fournir l'URL correspondant à l'entité du message. Pour les objets pouvant avoir plusieurs URL, un message de réponse peut inclure un en-tête Content-Location indiquant l'URL de l'objet utilisé pour générer la réponse. L'en-tête Content-Location peut être différent de l'URL demandée.

Ceci est généralement utilisé par les serveurs qui dirigent ou redirigent un client vers une nouvelle URL.

Si l'URL est relative, elle doit être interprétée par rapport à l'en-tête Content-Base. En l'absence de cet en-tête, l'URL utilisée dans la requête doit être utilisée.

En-tête d'entité de type

Syntaxe de base Contenu-Emplacement : (absoluteURL | relativeURL)

Exemple de contenu - Emplacement : http://www.joes-hardware.com/index.html

Contenu-MD5

L'en-tête Content-MD5 est utilisé par les serveurs pour vérifier l'intégrité du corps du message. Seul le serveur d'origine ou le client demandeur doit insérer un en-tête Content-MD5 dans le message. La valeur de l'en-tête est un condensé MD5* du corps du message (éventuellement codé).

La valeur de cet en-tête permet une vérification de bout en bout des données, utile pour détecter toute modification involontaire des données en transit. Son utilisation n'est pas destinée à des fins de sécurité.

La RFC 1864 définit cet en-tête plus en détail.

En-tête d'entité de type

Remarques La valeur de condensé MD5 est un condensé MD5 en base 64 (voir l'annexe E) ou 128 bits, tel que défini dans la RFC 1864.

Contenu de la syntaxe de base-MD5 : md5-digest

Exemple de contenu-MD5 : Q2h1Y2sgSW51ZwDIAXR5IQ==

Gamme de contenu

L'en-tête Content-Range est envoyé suite à une requête transmettant une plage d'un document. Il indique l'emplacement (plage) au sein de l'entité d'origine représentée par cette entité. Il indique également la longueur de l'entité entière.

Si un « * » est présent dans la valeur au lieu de la longueur de l'entité entière, cela signifie que la longueur n'était pas connue lors de l'envoi de la réponse. Consultez le chapitre 15 pour plus d'informations sur l'en-tête Content-Range.

En-tête d'entité de type

Les serveurs répondant avec le code de réponse 206 Contenu partiel ne doivent pas inclure d'en-tête Content-Range avec un « * » comme longueur.

Exemple de plage de contenu : octets 500-999 / 5 400

Type de contenu

L'en-tête Content-Type indique le type de média de l'objet dans le message.

En-tête d'entité de type

Syntaxe de base Content-Type : type de média

Exemple de type de contenu : text/html ; charset=iso-latin-1

* Le condensé MD5 est défini dans la RFC 1864.

Type de contenu

Biscuit

L'en-tête Cookie est un en-tête d'extension utilisé pour l'identification et le suivi du client.

Le chapitre 11 parle en détail de l’en-tête Cookie et de son utilisation (voir également « Set-Cookie »).

En-tête de demande d'extension de type

Exemple de cookie :
ink=IUOK164y59BC708378908CFF89OE5573998A115

Cookie2

L'en-tête Cookie2 est un en-tête d'extension utilisé pour l'identification et le suivi du client. Cookie2 permet d'identifier la version des cookies comprise par un demandeur. Il est défini plus en détail dans la RFC 2965.

Le chapitre 11 parle de l'en-tête Cookie2 et de son utilisation en détail.

En-tête de demande d'extension de type

Exemple Cookie2 : $version="1"

Date

L'en-tête « Date » indique la date et l'heure de création du message. Cet en-tête est obligatoire dans les réponses des serveurs, car l'heure et la date auxquelles le serveur estime que le message a été créé peuvent être utilisées par les caches pour évaluer la fraîcheur d'une réponse. Pour les clients, cet en-tête est totalement facultatif, bien qu'il soit de bon ton de l'inclure.

Type En-tête général

Syntaxe de base Date : HTTP-date

Exemples Date : mar. 3 oct. 1997 02:15:31 GMT

HTTP possède quelques formats de date spécifiques. Celui-ci est défini dans la RFC 822 et constitue le format privilégié pour les messages HTTP/1.1. Cependant, dans les spécifications HTTP antérieures, le format de date n'était pas précisé ; les implémenteurs serveur et client ont donc utilisé d'autres formats, qui doivent être pris en charge pour des raisons de sécurité. Vous rencontrerez des formats de date comme celui spécifié dans la RFC 850, ainsi que des dates au format produit par l'appel système asctime(). Les voici pour la date représentée ci-dessus :

> Date : mardi 3 oct. 97, 02:15:31 GMT, format RFC 850
>
> Date : mar. 3 oct. 02:15:31 1997 format asctime()

Le format asctime() est mal vu, car il utilise l'heure locale et ne précise pas le fuseau horaire (par exemple, GMT). En général, l'en-tête de date doit être en GMT ; cependant, les applications robustes doivent gérer les dates qui ne précisent pas le fuseau horaire ou qui incluent des valeurs de date en heure non GMT.

ETag

L'en-tête ETag fournit l'étiquette de l'entité contenue dans le message. Une étiquette d'entité permet d'identifier une ressource.

Les balises d'entité et leur relation avec les ressources sont décrites en détail au chapitre 15.

En-tête d'entité de type

Syntaxe de base ETag : entity-tag

Exemples ETag : « 11e92a-457b-31345aa »

Etag : W/"11e92a-457b-3134b5aa"

Attendre

L'en-tête « Expect » est utilisé par les clients pour informer les serveurs qu'ils attendent un comportement particulier. Cet en-tête est actuellement étroitement lié au code de réponse « Continue » (voir « 100–199 : Codes d'état informatifs » au chapitre 3).

Si un serveur ne comprend pas la valeur de l'en-tête Expect, il doit répondre avec un code d'état 417 Expectation Failed.

Type d'en-tête de demande

Syntaxe de base Attendre : 1# (« 100-continuer » | attente-extension)

Exemple : Attendre : 100 - continuer

Expire

L'en-tête Expires indique la date et l'heure auxquelles la réponse n'est plus valide. Cela permet aux clients, comme votre navigateur, de mettre une copie en cache et de ne pas avoir à demander au serveur si elle est toujours valide avant l'expiration de ce délai.

Le chapitre 7 explique comment l'en-tête Expires est utilisé, en particulier comment il est lié aux caches et à la nécessité de revalider les réponses avec le serveur d'origine.

En-tête d'entité de type

Syntaxe de base Expires : HTTP-date

Exemple : Expire le : jeu. 3 oct. 1997 17:15:00 GMT

Depuis

L'en-tête « De » indique l'expéditeur de la requête. Le format correspond simplement à une adresse e-mail Internet valide (spécifiée dans la RFC 1123) pour l'utilisateur du client.

Depuis

L'utilisation et le remplissage de cet en-tête peuvent poser des problèmes de confidentialité. Les implémenteurs clients doivent veiller à informer leurs utilisateurs et à leur donner le choix avant d'inclure cet en-tête dans un message de requête. Compte tenu du risque d'abus de la part de personnes collectant des adresses e-mail pour des messages non sollicités, malheur à l'implémenteur qui diffuse cet en-tête sans prévenir et doit répondre à des utilisateurs mécontents.

Type d'en-tête de demande

Syntaxe de base De : boîte aux lettres

Exemple de : slurp@inktomi.com

Hôte

L'en-tête Host est utilisé par les clients pour fournir au serveur le nom d'hôte et le numéro de port Internet de la machine depuis laquelle le client souhaite effectuer une requête. Le nom d'hôte et le port correspondent à l'URL demandée par le client.

L'en-tête Host permet aux serveurs de différencier différentes URL relatives en fonction du nom d'hôte, donnant au serveur la possibilité d'héberger plusieurs noms d'hôtes différents sur la même machine.

(c'est-à-dire la même adresse IP).

Type d'en-tête de demande

Remarques : les clients HTTP/1.1 doivent inclure un en-tête Host dans toutes leurs requêtes. Tous les serveurs HTTP/1.1 doivent répondre avec le code d'état 400 Bad Request.

Clients HTTP/1.1 qui ne fournissent pas d'en-tête Host.

Syntaxe de base Hôte : hôte [":" port]

Exemple d'hôte : www.hotbot.com:80

Si-Modifié-Depuis

L'en-tête de requête If-Modified-Since permet d'effectuer des requêtes conditionnelles. Un client peut utiliser la méthode GET pour demander une ressource à un serveur, la réponse dépendant si la ressource a été modifiée depuis sa dernière demande.

Si l'objet n'a pas été modifié, le serveur répondra par une réponse 304 « Non modifié », au lieu de la ressource. Si l'objet a été modifié, le serveur répondra comme s'il s'agissait d'une requête GET non conditionnelle. Le chapitre 7 détaille les requêtes conditionnelles.

Type d'en-tête de demande

Syntaxe de base If-Modified-Since : HTTP-date

Exemple : Si modifié depuis : jeu. 3 oct. 1997 17:15:00 GMT

Si-Match

Tout comme l'en-tête If-Modified-Since, l'en-tête If-Match permet de conditionner une requête. Au lieu d'une date, la requête If-Match utilise une balise d'entité. Le serveur compare la balise d'entité de l'en-tête If-Match à la balise d'entité actuelle de la ressource et renvoie l'objet si les balises correspondent.

Le serveur doit utiliser la valeur If-Match de « * » pour faire correspondre toute balise d'entité dont il dispose pour une ressource ; « * » correspondra toujours, à moins que le serveur ne dispose plus de la ressource.

Cet en-tête est utile pour mettre à jour les ressources déjà présentes dans un client ou un cache. La ressource est renvoyée uniquement si elle a changé, c'est-à-dire si la balise d'entité de l'objet précédemment demandé ne correspond pas à celle de la version actuelle sur le serveur. Le chapitre 7 détaille les requêtes conditionnelles.

Type d'en-tête de demande

Syntaxe de base If-Match : (« * » | 1 # entité-balise)

Exemple de correspondance : « 11e92a-457b-31345aa »

Si-Aucune-Correspondance

L'en-tête If-None-Match, comme tous les en-têtes If, permet de conditionner une requête. Le client fournit au serveur une liste de balises d'entité, et le serveur compare ces balises à celles de la ressource, ne renvoyant la ressource que si aucune ne correspond.

Cela permet à un cache de mettre à jour les ressources uniquement si elles ont changé. Grâce à l'en-tête If-NoneMatch, un cache peut utiliser une seule requête pour invalider les entités qu'il possède et recevoir la nouvelle entité dans la réponse. Le chapitre 7 détaille les requêtes conditionnelles.

Type d'en-tête de demande

Syntaxe de base If-None-Match : (« * » | 1 # entité-balise)

Exemple If-None-Match : « 11e92a-457b-31345aa »

Plage If

L'en-tête If-Range, comme tous les en-têtes If, permet de rendre une requête conditionnelle. Il est utilisé lorsqu'une application possède une copie d'une plage de ressources, pour revalider la plage ou obtenir la ressource complète si la plage n'est plus valide. Le chapitre 7 détaille les requêtes conditionnelles.

Type d'en-tête de demande

Syntaxe de base If-Range : (date HTTP | balise d'entité)

Plage If

Exemples If-Range : mar. 3 oct. 1997 02:15:31 GMT If-Range : "11e92a-457b-3134b5aa"

Si-Non-Modifié-Depuis

L'en-tête If-Unmodified-Since est le jumeau de l'en-tête If-Modified-Since. Son inclusion dans une requête rend celle-ci conditionnelle. Le serveur doit examiner la valeur de date de l'en-tête et renvoyer l'objet uniquement s'il n'a pas été modifié depuis la date indiquée.

Le chapitre 7 traite en détail des demandes conditionnelles.

Type d'en-tête de demande

Syntaxe de base If-Unmodified-Since : HTTP-date

Exemple : Si-Non modifié-Depuis : Jeu, 03 oct. 1997 17:15:00 GMT

Dernière modification

L'en-tête Last-Modified tente de fournir des informations sur la dernière modification de cette entité. Cela peut avoir plusieurs significations. Par exemple, les ressources sont généralement des fichiers sur un serveur ; la valeur Last-Modified peut donc correspondre à l'heure de dernière modification fournie par le système de fichiers du serveur. En revanche, pour les ressources créées dynamiquement, comme celles créées par des scripts, la valeur Last-Modified peut correspondre à l'heure de création de la réponse.

Les serveurs doivent veiller à ce que l'heure de dernière modification ne soit pas ultérieure. Les serveurs HTTP/1.1 doivent réinitialiser l'heure de dernière modification si elle est postérieure à la valeur envoyée dans l'en-tête Date.

En-tête d'entité de type

Syntaxe de base Dernière modification : HTTP-date

Exemple Dernière modification : jeu. 03 oct. 1997 17:15:00 GMT

Emplacement

L'en-tête Location est utilisé par les serveurs pour diriger les clients vers l'emplacement d'une ressource qui a été déplacée depuis la dernière demande du client ou qui a été créée en réponse à la demande.

Type d'en-tête de réponse

Emplacement de la syntaxe de base : absoluteURL

Exemple d'emplacement : http://www.hotbot.com

Max-Forwards

Cet en-tête est utilisé uniquement avec la méthode TRACE, afin de limiter le nombre de proxys ou autres intermédiaires par lesquels transite une requête. Sa valeur est un entier. Chaque application recevant une requête TRACE avec cet en-tête doit décrémenter cette valeur avant de la transmettre.

Si la valeur est nulle lorsque l'application reçoit la requête, elle doit renvoyer une réponse 200 OK, avec un corps d'entité contenant la requête d'origine. Si l'en-tête MaxForwards est absent d'une requête TRACE, il faut supposer qu'il n'existe pas de nombre maximal de transferts.

Pour les autres méthodes HTTP, cet en-tête doit être ignoré. Consultez la section « Méthodes » du chapitre 3 pour en savoir plus sur la méthode TRACE.

Type d'en-tête de demande

Syntaxe de base Max-Forwards : 1*DIGIT

Exemple Max-Forwards : 5

Version MIME

MIME est le cousin de HTTP. Bien que radicalement différents, certains serveurs HTTP génèrent des messages valides selon la spécification MIME. Dans ce cas, l'en-tête MIME-Version peut être fourni par le serveur.

Cet en-tête n'a jamais fait partie de la spécification officielle, bien qu'il soit mentionné dans la spécification HTTP/1.0. De nombreux serveurs plus anciens envoient des messages avec cet en-tête. Cependant, ces

messages ne sont souvent pas des messages MIME valides, ce qui rend cet en-tête à la fois confus et indigne de confiance.

Extension de type en-tête général

Syntaxe de base Version MIME : DIGIT "." DIGIT

Exemple de version MIME : 1.0

Pragma

L'en-tête Pragma sert à transmettre des instructions avec le message. Ces instructions peuvent être de nature très variée, mais elles servent souvent à contrôler le comportement de mise en cache. Les proxys et les passerelles ne doivent pas supprimer l'en-tête Pragma, car il pourrait être destiné à toutes les applications recevant le message.

La forme la plus courante de Pragma, Pragma: no-cache, est un en-tête de requête qui force les caches à demander ou à revalider le document auprès du serveur d'origine, même lorsqu'une nouvelle copie est disponible dans le cache. Il est envoyé par les navigateurs lorsque les utilisateurs cliquent sur le bouton Recharger/Actualiser. De nombreux serveurs envoient Pragma: no-cache comme en-tête de réponse (équivalent de

Pragma

Cache-Control : no-cache), mais malgré son utilisation courante, ce comportement est techniquement indéfini. Toutes les applications ne prennent pas en charge les en-têtes de réponse Pragma.

Le chapitre 7 décrit l'en-tête Pragma et la manière dont il est utilisé par les applications HTTP/1.0 pour contrôler les caches.

Type d'en-tête de demande

Syntaxe de base Pragma : 1# pragma-directive*

Exemple de pragma : no-cache

Proxy-Authentifier

L'en-tête Proxy-Authenticate fonctionne comme l'en-tête WWW-Authenticate. Il est utilisé par les proxys pour demander à une application qui envoie une requête de s'authentifier. Les détails de ce défi/réponse, ainsi que d'autres mécanismes de sécurité HTTP, sont détaillés au chapitre 14.

Si un serveur proxy HTTP/1.1 envoie une réponse 407 Authentification proxy requise, il doit inclure l'en-tête Proxy-Authenticate.

Les proxys et les passerelles doivent interpréter avec prudence les en-têtes de proxy. Il s'agit généralement d'en-têtes segmentés, s'appliquant uniquement à la connexion en cours. Par exemple, l'en-tête ProxyAuthenticate demande une authentification pour la connexion en cours.

Type d'en-tête de réponse

Syntaxe de base Proxy-Authenticate : défi

Exemple d'authentification par proxy : domaine de base = « Documents financiers d'entreprise ultra-secrets »

Autorisation de procuration

L'en-tête Proxy-Authorization fonctionne comme l'en-tête Authorization. Il est utilisé par les applications clientes pour répondre aux défis Proxy-Authenticate. Consultez le chapitre 14 pour en savoir plus sur le fonctionnement du mécanisme de sécurité défi/réponse.

Type d'en-tête de demande

Syntaxe de base Proxy-Authorization : informations d'identification

Exemple d'autorisation de proxy : Basic YnJpYW4tdG90dHk6T3ch

* La seule directive Pragma définie par la spécification est « no-cache » ; cependant, vous pouvez rencontrer d'autres en-têtes Pragma qui ont été définis comme des extensions de la spécification.

Connexion proxy

L'en-tête Proxy-Connection devait avoir une sémantique similaire à celle de l'en-tête HTTP/1.0 Connection. Il devait être utilisé entre les clients et les proxys pour spécifier les options relatives aux connexions (principalement les connexions persistantes).* Ce n'est pas un en-tête standard et il est considéré comme un en-tête ad hoc par le comité de normalisation. Cependant, il est largement utilisé par les navigateurs et les proxys.

Les développeurs de navigateurs ont créé l'en-tête Proxy-Connection pour résoudre le problème d'un client envoyant un en-tête de connexion HTTP/1.0 qui est transmis à l'aveugle par un proxy passif. Un serveur recevant cet en-tête de connexion pourrait confondre les fonctionnalités de la connexion client avec celles de la connexion proxy.

L'en-tête Proxy-Connection est envoyé à la place de l'en-tête Connection lorsque le client sait qu'il transite par un proxy. Les serveurs ne reconnaissant pas l'en-tête ProxyConnection, ils l'ignorent, permettant ainsi aux proxys passifs de le transmettre aveuglément sans causer de dommages.

Le problème avec cette solution survient s'il y a plusieurs proxys sur le chemin du client vers le serveur. Si le premier transmet aveuglément l'en-tête au second, qui le comprend, ce dernier risque de subir la même confusion que le serveur avec l'en-tête de connexion.

C'est le problème rencontré par le groupe de travail HTTP avec cette solution : ils y ont vu une astuce permettant de résoudre le problème d'un proxy unique, mais pas le problème principal. Néanmoins, elle gère certains des cas les plus courants, et comme les anciennes versions de Netscape Navigator et de Microsoft Internet Explorer l'implémentent, les développeurs de proxy doivent s'en occuper. Voir le chapitre 4 pour plus d'informations.

Type En-tête général

Syntaxe de base Proxy-Connection : 1# (jeton de connexion)

Exemple de connexion proxy : fermer

Publique

L'en-tête Public permet à un serveur d'indiquer à un client les méthodes qu'il prend en charge. Ces méthodes pourront être utilisées par le client lors de futures requêtes. Les proxys doivent être vigilants lorsqu'ils reçoivent une réponse d'un serveur contenant l'en-tête Public. L'en-tête indique les capacités du serveur, et non du proxy. Ce dernier doit donc modifier la liste des méthodes dans l'en-tête ou supprimer l'en-tête avant d'envoyer la réponse au client.

Type d'en-tête de réponse

* Voir le chapitre 4 pour en savoir plus sur les connexions persistantes et persistantes.

Publique

Remarques Cet en-tête n'est pas défini dans la RFC 2616. Il était précédemment défini dans la RFC 2068, une version antérieure de la spécification HTTP/1.1, mais il a depuis été supprimé de la spécification officielle.

Syntaxe de base Public : 1# méthode HTTP

Exemple Public : OPTIONS, GET, HEAD, TRACE, POST

Gamme

L'en-tête Range est utilisé dans les requêtes portant sur des parties ou des plages d'une entité. Sa valeur indique la plage de l'entité incluse dans le message.

Les requêtes portant sur des plages d'un document permettent de demander plus efficacement des objets volumineux (en les demandant par segments) ou de récupérer des données après un échec de transfert (en permettant à un client de demander la plage de la ressource non atteinte). Les requêtes de plage et les en-têtes qui les rendent possibles sont détaillés au chapitre 15.

En-tête d'entité de type

Exemple de plage : octets = 500-1 500

Référent

L'en-tête Referer est inséré dans les requêtes client pour indiquer au serveur d'où provient l'URL. Il s'agit d'une démarche volontaire, au bénéfice du serveur ; elle lui permet de mieux enregistrer les requêtes ou d'effectuer d'autres tâches. L'orthographe erronée de « Referer » rappelle les débuts de HTTP, au grand dam des réviseurs anglophones du monde entier.

Le fonctionnement de votre navigateur est assez simple. Si vous accédez à la page d'accueil A et cliquez sur un lien pour accéder à la page d'accueil B, votre navigateur insère un en-tête Referer dans la requête portant la valeur A. Les en-têtes Referer sont insérés par votre navigateur uniquement lorsque vous cliquez sur des liens ; les requêtes d'URL que vous saisissez vous-même ne contiennent pas d'en-tête Referer.

Certaines pages étant privées, cet en-tête soulève des problèmes de confidentialité. Bien que cela relève d'une paranoïa injustifiée, il permet aux serveurs web et à leurs administrateurs de savoir d'où vous venez, ce qui leur permet potentiellement de mieux suivre votre navigation. Par conséquent, la spécification HTTP/1.1 recommande aux développeurs d'applications de laisser l'utilisateur décider si cet en-tête doit être transmis.

Type d'en-tête de demande

Syntaxe de base Référent : (absoluteURL | relativeURL)

Exemple de référent : http://www.inktomi.com/index.html

Réessayer après

Les serveurs peuvent utiliser l'en-tête Retry-After pour indiquer au client quand relancer sa requête de ressource. Il est utilisé avec le code d'état 503 Service indisponible pour indiquer au client une date et une heure précises (ou un nombre de secondes) auxquelles il doit relancer sa requête.

Un serveur peut également utiliser cet en-tête lorsqu'il redirige les clients vers des ressources, donnant au client un temps d'attente avant

de faire une demande sur la ressource vers laquelle il est redirigé.* Cela peut être très utile pour les serveurs qui créent des ressources dynamiques, permettant au serveur de rediriger le client vers la ressource nouvellement créée mais donnant le temps à la ressource d'être créée.

Type d'en-tête de réponse

Syntaxe de base Retry-After : (date HTTP | delta-secondes)

Exemples Réessayer après : mar. 3 oct. 1997 02:15:31 GMT Réessayer après : 120

Serveur

L'en-tête Serveur est similaire à l'en-tête Agent Utilisateur ; il permet aux serveurs de s'identifier auprès des clients. Sa valeur est le nom du serveur et un commentaire facultatif le concernant.

Étant donné que l'en-tête Serveur identifie le produit serveur et peut contenir des commentaires supplémentaires, son format est relativement libre. Si vous développez un logiciel qui dépend de l'identification d'un serveur, il est conseillé de tester le logiciel serveur pour voir ce qu'il renvoie, car ces jetons varient d'un produit à l'autre et d'une version à l'autre.

Comme avec l'en-tête User-Agent, ne soyez pas surpris si un ancien proxy ou une ancienne passerelle insère ce qui équivaut à un en-tête Via dans l'en-tête Server lui-même.

Type d'en-tête de réponse

Serveur de syntaxe de base : 1* (produit | commentaire)

Exemples Serveur : Microsoft-Internet-Information-Server/1.0

Serveur : websitepro/1.1f (s/n wpo-07d0)

Serveur : apache/1.2b6 via une passerelle proxy CERN-HTTPD/3.0 libwww/2.13

Set-Cookie

L'en-tête Set-Cookie est le partenaire de l'en-tête Cookie ; dans le chapitre 11, nous discutons en détail de l'utilisation de cet en-tête.

* Voir « Codes d’état de redirection et phrases de raison » au chapitre 3 pour plus d’informations sur les réponses de redirection du serveur.

Set-Cookie

En-tête de réponse d'extension de type

Syntaxe de base Set-Cookie : commande

Exemples Set-Cookie : lastorder=00183 ; path=/orders Set-Cookie : private_id=519 ; secure

Set-Cookie2

L'en-tête Set-Cookie2 est une extension de l'en-tête Set-Cookie ; dans le chapitre 11, nous discutons en détail de l'utilisation de cet en-tête.

En-tête de réponse d'extension de type

Syntaxe de base Set-Cookie2 : commande

Exemples Set-Cookie2 : ID="29046"; Domain=".joes-hardware.com" Set-Cookie2 : color=blue

TE

L'en-tête TE, mal nommé, fonctionne comme l'en-tête Accept-Encoding, mais pour les encodages de transfert (il aurait pu s'appeler Accept-Transfer-Encoding, mais ce n'est pas le cas). L'en-tête TE peut également être utilisé pour indiquer si un client peut gérer les en-têtes de la fin d'une réponse ayant subi l'encodage fragmenté. Consultez le chapitre 15 pour plus d'informations sur l'en-tête TE, l'encodage fragmenté et les fins de réponse.

Type d'en-tête de demande

Remarques : si la valeur est vide, seul le codage de transfert fragmenté est acceptable. Le jeton spécial « trailers » indique que les en-têtes de fin sont acceptables dans une réponse fragmentée.

Syntaxe de base TE : # (codages de transfert)

transfer-codings= "remorques" | (transfer-extension [accept-params])

Exemples TE :

TE : en morceaux

Bande-annonce

L'en-tête Trailer est utilisé pour indiquer quels en-têtes sont présents dans la bande-annonce d'un message.

Le chapitre 15 traite en détail des encodages fragmentés et des bandes-annonces.

Type En-tête général

Syntaxe de base : 1#nom-de-champ

Exemple de bande-annonce : durée du contenu

Titre

L'en-tête « Titre » est un en-tête non spécifié censé indiquer le titre de l'entité. Cet en-tête faisait partie d'une ancienne extension HTTP/1.0 et était principalement utilisé pour les pages HTML, qui disposent de marqueurs de titre clairs utilisables par les serveurs. Comme de nombreux types de médias sur le Web, voire la plupart, ne permettent pas d'extraire un titre aussi facilement, cet en-tête a une utilité limitée. De ce fait, il n'a jamais été intégré à la spécification officielle, bien que certains serveurs plus anciens le transmettent encore fidèlement.

Type d'en-tête de réponse

Remarques L'en-tête Title n'est pas défini dans la RFC 2616. Il a été défini à l'origine dans le projet de définition HTTP/1.0 (http://www.w3.org/Protocols/HTTP/HTTP2.html) mais a depuis été supprimé de la spécification officielle.

Syntaxe de base Titre : document-title

Exemple de titre : CNN Interactive

Transfert-Encodage

Si un codage a dû être effectué pour transférer le corps du message HTTP en toute sécurité, le message contiendra l'en-tête Transfer-Encoding. Sa valeur est la liste des codages appliqués au corps du message. Si plusieurs codages ont été appliqués, ils sont listés dans l'ordre.

L'en-tête Transfer-Encoding diffère de l'en-tête Content-Encoding car l'encodage de transfert est un encodage effectué par un serveur ou une autre application intermédiaire pour transférer le message.

Les codages de transfert sont abordés au chapitre 15.

Type En-tête général

Syntaxe de base Transfert-Encodage : 1# transfer-coding

Exemple de transfert-codage : fragmenté

UA- (CPU, Disp, OS, Couleur, Pixels)

Ces en-têtes User-Agent ne sont pas standard et ne sont plus courants. Ils fournissent des informations sur la machine cliente, ce qui pourrait permettre une meilleure sélection du contenu par un serveur.

UA- (CPU, Disp, OS, Couleur, Pixels)

Par exemple, si un serveur savait que la machine d'un utilisateur n'avait qu'un écran couleur 8 bits, le serveur pourrait sélectionner des images optimisées pour ce type d'affichage.

Avec tout en-tête qui fournit des informations sur le client qui autrement ne seraient pas disponibles, il existe des problèmes de sécurité (voir le chapitre 14 pour plus d'informations).

En-têtes de demande d'extension de type

Remarques Ces en-têtes ne sont pas définis dans la RFC 2616 et leur utilisation est mal vue.

Syntaxe de base "UA" "-" ("CPU" | "Affichage" | "SE" | "Couleur" | "Pixels") ":" valeur-machine valeur-machine = (cpu | taille-écran | nom-système-d'exploitation | profondeur-de-couleur-d'affichage)

Exemples UA-CPU : CPU x86 de la machine du client

UA-Disp : 640, 480, 8 Taille et profondeur de couleur de l'écran du client

UA-OS : système d'exploitation Windows 95 de la machine cliente

UA-Color : color8 Profondeur de couleur de l'écran du client

UA-Pixels : 640 x 480 Taille de l'écran du client

Mise à niveau

L'en-tête Upgrade permet à l'expéditeur d'un message d'indiquer son souhait d'utiliser un autre protocole, voire un protocole complètement différent. Par exemple, un client HTTP/1.1 pourrait envoyer une requête HTTP/1.0 à un serveur et inclure un en-tête Upgrade avec la valeur « HTTP/1.1 », permettant ainsi au client de tâter le terrain et de vérifier si le serveur utilise HTTP/1.1.

Si le serveur le permet, il peut envoyer une réponse appropriée indiquant au client qu'il peut utiliser le nouveau protocole. Cela constitue un moyen efficace de migrer vers d'autres protocoles. La plupart des serveurs sont actuellement compatibles uniquement avec HTTP/1.0, et cette stratégie permet au client d'éviter de perturber un serveur avec trop d'en-têtes HTTP/1.1 jusqu'à ce qu'il détermine si le serveur est effectivement capable de communiquer avec HTTP/1.1.

Lorsqu'un serveur envoie une réponse 101 Switching Protocols, il doit inclure cet en-tête.

Type En-tête général

Mise à niveau de la syntaxe de base : protocole 1#

Exemple de mise à niveau : HTTP/2.0

Agent utilisateur

L'en-tête User-Agent est utilisé par les applications clientes pour s'identifier, à l'instar de l'en-tête Server pour les serveurs. Sa valeur est le nom du produit et éventuellement un commentaire décrivant l'application cliente.

Le format de cet en-tête est relativement libre. Sa valeur varie selon le produit client et la version. Il contient parfois même des informations sur la machine sur laquelle le client est exécuté.

Comme pour l'en-tête du serveur, ne soyez pas surpris si les anciennes applications proxy ou passerelle insèrent ce qui équivaut à un en-tête Via dans l'en-tête User-Agent lui-même.

Type ; en-tête de la requête

Syntaxe de base User-Agent : 1* (produit | commentaire)

Exemple d'agent utilisateur : Mozilla/4.0 (compatible ; MSIE 5.5 ; Windows NT 5.0)

Varier

L'en-tête Vary est utilisé par les serveurs pour informer les clients quels en-têtes de la demande d'un client seront utilisés dans la négociation côté serveur.* Sa valeur est une liste d'en-têtes que le serveur examine pour déterminer ce qu'il faut envoyer au client en guise de réponse.

Prenons l'exemple d'un serveur qui envoie des pages HTML spécifiques en fonction des fonctionnalités de votre navigateur web. Un serveur envoyant ces pages spécifiques pour une URL inclurait un en-tête Vary indiquant qu'il a examiné l'en-tête User-Agent de la requête pour déterminer la réponse à envoyer.

L'en-tête Vary est également utilisé par les proxys de mise en cache ; voir le chapitre 7 pour en savoir plus sur la relation entre l'en-tête Vary et les réponses HTTP mises en cache.

Type d'en-tête de réponse

Syntaxe de base variable : (« * » | 1 # nom-de-champ)

Exemple de variation : agent utilisateur

Via

L'en-tête Via permet de suivre les messages transitant par les proxys et les passerelles. Cet en-tête informatif permet de savoir quelles applications traitent les requêtes et les réponses.

Lorsqu'un message transite par une application HTTP pour atteindre un client ou un serveur, cette application peut utiliser l'en-tête Via pour marquer le message comme ayant transité par elle. Il s'agit d'un en-tête HTTP/1.1 ; de nombreuses applications plus anciennes insèrent une chaîne de type Via dans les en-têtes User-Agent ou Server des requêtes et des réponses.

Si le message transite par plusieurs applications intermédiaires, chacune d'elles doit ajouter sa chaîne Via. L'en-tête Via doit être inséré par les proxys et passerelles HTTP/1.1.

* Voir le chapitre 17 pour plus d'informations sur la négociation de contenu.

Via

En-tête général

* Via : 1# (protocole reçu reçu par [commentaire])

Via : 1.1 joes-hardware.com (Joes-Server/1.0)

Ce qui précède indique que le message a transité par le logiciel Joes Server version 1.0 exécuté sur la machine joes-hardware.com. Le serveur Joe utilisait HTTP 1.1. L'en-tête Via doit être formaté comme suit : HTTP-Version machine-hostname (Application-Name-Version)

Avertissement

L'en-tête d'avertissement permet de fournir des informations complémentaires sur le déroulement d'une requête. Il permet au serveur d'envoyer des informations supplémentaires qui ne figurent pas dans le code d'état ou la phrase de raison. Plusieurs codes d'avertissement sont définis dans la spécification HTTP/1.1 :

La réponse 101 est obsolète

Lorsqu'un message de réponse est connu comme étant obsolète (par exemple, si le serveur d'origine n'est pas disponible pour une revalidation), cet avertissement doit être inclus.

111 Échec de la revalidation

Si un cache tente de revalider une réponse avec un serveur d'origine et que la revalidation échoue parce que le cache ne peut pas atteindre le

serveur d'origine, cet avertissement doit être inclus dans la réponse au client.

112 Fonctionnement déconnecté

Un avertissement informatif ; doit être utilisé si la connectivité d'un cache au réseau est supprimée.

113 Expiration heuristique

Les caches doivent inclure cet avertissement si leur heuristique de fraîcheur est supérieure à 24 heures et qu'ils renvoient une réponse avec un âge supérieur à 24 heures.

199 Avertissement divers

Les systèmes recevant cet avertissement ne doivent pas accepter de réponse automatisée ; le message peut et doit probablement contenir un corps avec des informations supplémentaires pour l'utilisateur.

214 Transformation appliquée

Doit être ajouté par toute application intermédiaire, telle qu'un proxy, si l'application effectue une transformation qui modifie l'encodage du contenu de la réponse.

299 Avertissement persistant divers

Les systèmes recevant cet avertissement ne doivent pas prendre de réaction automatisée ; l'erreur peut contenir un corps avec plus d'informations pour l'utilisateur.

* Consultez la spécification HTTP/1.1 pour la syntaxe complète de l'en-tête Via.

En-tête de réponse

Avertissement : 1 # valeur-avertissement

Avertissement : 113

WWW-Authentifier

L'en-tête WWW-Authenticate est utilisé dans les réponses 401 non autorisées pour émettre un schéma d'authentification par défi au client. Le chapitre 14 décrit l'en-tête WWW-Authenticate et son utilisation dans le système d'authentification par défi/réponse de base de HTTP.

Type d'en-tête de réponse

Syntaxe de base WWW-Authenticate : 1# challenge

Exemple d'authentification WWW : domaine de base = « Votre profil de voyage privé »

X-Cache

Les en-têtes X sont tous des en-têtes d'extension. L'en-tête X-Cache est utilisé par Squid pour informer un client de la disponibilité d'une ressource.

En-tête de réponse d'extension de type

Exemple X-Cache : HIT

X-Transféré-Pour

Cet en-tête est utilisé par de nombreux serveurs proxy (par exemple, Squid) pour indiquer à qui une requête a été transmise. Comme l'en-tête Client-ip mentionné précédemment, cet en-tête de requête indique l'adresse d'origine de la requête.

En-tête de demande d'extension de type

Syntaxe de base X-Forwarded-For : addr

Exemple X-Forwarded-For : 64.95.76.161

X-Pad

Cet en-tête est utilisé pour surmonter un bogue lié à la longueur de l'en-tête de réponse dans certains navigateurs ; il remplit les en-têtes des messages de réponse avec des octets supplémentaires pour contourner le bogue.

X-Pad

En-tête général de l'extension

X-Pad : pad-texte

X-Pad : bogosity

Numéro de série X

L'en-tête X-Serial-Number est un en-tête d'extension. Il était utilisé par certaines anciennes applications HTTP pour insérer le numéro de série du logiciel sous licence dans le message HTTP.

Son utilisation a pratiquement disparu, mais il est répertorié ici comme un exemple des en-têtes X qui existent.

Extension de type en-tête général

Syntaxe de base X-Serial-Number : serialno

Exemple de numéro de série X : 010014056

Annexe DCeci est le titre du livre ANNEXE D

Types MIME

Les types de médias MIME (types MIME, en abrégé) sont des noms normalisés décrivant le contenu d'une entité de message (par exemple, texte/html, image/jpeg). Cette annexe explique le fonctionnement des types MIME, comment en enregistrer de nouveaux et où trouver plus d'informations.

De plus, cette annexe contient dix tableaux pratiques détaillant des centaines de types MIME, issus de nombreuses sources du monde entier. Il s'agit probablement de la liste tabulaire la plus détaillée jamais compilée. Nous espérons que ces tableaux vous seront utiles.

Dans cette annexe, nous allons :

• Décrivez le matériel de référence principal, dans « Contexte ».

• Expliquez la structure des types MIME dans « Structure du type MIME ». • Montrez comment enregistrer les types MIME dans « Enregistrement IANA du type MIME ».

• Facilitez la recherche des types MIME.

Les tables de types MIME suivantes sont incluses dans cette annexe :

• application/*—Tableau D-3

• audio/*—Tableau D-4

• chimique/*—Tableau D-5

• image/*—Tableau D-6

• message/*—Tableau D-7

• modèle/*—Tableau D-8

• multipart/*—Tableau D-9

• texte/*—Tableau D-10

• vidéo/*—Tableau D-11

• Autre — Tableau D-12

Arrière-plan

Les types MIME ont été développés à l'origine pour le courrier électronique multimédia (MIME signifie Multipurpose Internet Mail Extensions), mais ils ont été réutilisés pour HTTP et plusieurs autres

protocoles qui doivent décrire le format et l'objectif des objets de données.

MIME est défini par cinq documents principaux :

RFC 2045, « MIME : format des corps de messages Internet »

Décrit la structure globale du message MIME et présente l'en-tête ContentType, emprunté par HTTP

RFC 2046, « MIME : types de médias »

Présente les types MIME et leur structure

RFC 2047, « MIME : extensions d'en-tête de message pour le texte non-ASCII » Définit les moyens d'inclure des caractères non-ASCII dans les en-têtes

RFC 2048, « MIME : procédures d'enregistrement »

Définit comment enregistrer les valeurs MIME auprès de l'Internet Assigned Numbers Authority (IANA)

RFC 2049, « MIME : critères de conformité et exemples »

Détaille les règles de conformité et fournit des exemples

Aux fins de HTTP, nous sommes particulièrement intéressés par la RFC 2046 (types de médias) et

RFC 2048 (Procédures d'enregistrement).

Structure du type MIME

Chaque type de média MIME se compose d'un type, d'un sous-type et d'une liste de paramètres facultatifs. Le type et le sous-type sont séparés par une barre oblique, et les paramètres facultatifs commencent par un point-virgule, s'ils sont présents. En HTTP, les types de média MIME sont largement utilisés dans les en-têtes Content-Type et Accept. Voici quelques exemples :

Type de contenu : vidéo/quicktime

Type de contenu : texte/html ; jeu de caractères = iso-8859-6

Type de contenu : multipart/mixte ; boundary=gc0p4Jq0M2Yt08j34c0p

Accepter : image/gif

Types discrets

Les types MIME peuvent décrire directement le type d'objet, ou décrire des collections ou des packages d'autres types d'objets. Si un type MIME décrit directement un type d'objet, il s'agit d'un type discret. Il s'agit notamment des fichiers texte, des vidéos et des formats de fichiers spécifiques à une application.

Types composites

Si un type MIME décrit une collection ou une encapsulation d'un autre contenu, il est appelé type composite. Un type composite décrit le format du package englobant. À l'ouverture du package englobant, chaque objet englobé possède son propre type.

Types en plusieurs parties

Les types de médias multiparties sont des types composites. Un objet multipartie est constitué de plusieurs types de composants. Voici un exemple de contenu multipartie/mixte, où chaque composant possède son propre type MIME :

Type de contenu : multipart/mixte ; boundary=unique-boundary-1

--limite-unique-1

Type de contenu : texte/plain ; jeu de caractères = US-ASCII Salut, je suis un texte ASCII ennuyeux...

--limite-unique-1

Type de contenu : multipart/parallèle ; boundary=unique-boundary-2

--limite-unique-2

Type de contenu : audio/basique

... Les données audio au format mu-law monocanal 8000 Hz vont ici...

--limite-unique-2

Type de contenu : image/jpeg

... les données d'image vont ici ...

--limite-unique-2--

--limite-unique-1

Type de contenu : texte/enrichi

Ceci est <bold><italic>enrichi.</italic></bold>

<smaller>comme défini dans la RFC 1896</smaller> N'est-ce pas <bigger><bigger>cool ?</bigger></bigger>

--limite-unique-1

Type de contenu : message/rfc822

De : (boîte aux lettres en US-ASCII)

À : (adresse en US-ASCII)

Objet : (sujet en US-ASCII)

Type de contenu : Texte brut ; jeu de caractères = ISO-8859-1

Codage de transfert de contenu : imprimable entre guillemets... Le texte supplémentaire dans ISO-8859-1 va ici...

--limite-unique-1--

Structure du type MIME

Syntaxe

Comme nous l'avons indiqué précédemment, les types MIME se composent d'un type principal, d'un sous-type et d'une liste facultative de paramètres.

Le type principal peut être un type prédéfini, un jeton d'extension défini par l'IETF ou un jeton expérimental (commençant par « x- »). Certains types principaux courants sont décrits dans le tableau D-1.

Tableau D-1. Types MIME principaux courants

Description du type

format de contenu spécifique à l'application (type discret)

audio Format audio (type discret)

Ensemble de données chimiques (type d'extension IETF discret)

image Format d'image (type discret)

message Format du message (type composite)

modèle format de modèle 3D (type d'extension IETF discret)

Collection multipart de plusieurs objets (type composite)

texte Format de texte (type discret)

vidéo Format de film vidéo (type discret)

Les sous-types peuvent être des types primaires (comme dans « texte/texte »), des sous-types enregistrés auprès de l'IANA ou des jetons d'extension expérimentaux (commençant par « x- »).

Les types et sous-types sont constitués d'un sous-ensemble de caractères US-ASCII. Les espaces et certains caractères de groupement et de ponctuation réservés, appelés « tspecials », sont des caractères de contrôle et sont interdits dans les noms de types et de sous-types.

La grammaire de la RFC 2046 est présentée ci-dessous :

TYPE := "application" | "audio" | "image" | "message" | "multipart" |

« texte » | « vidéo » | IETF-TOKEN | X-TOKEN

SOUS-TYPE := SOUS-JETON IANA | JETON IETF | JETON X

IETF-TOKEN := <jeton d'extension avec RFC et enregistré auprès de l'IANA>

IANA-SUBTOKEN := <jeton d'extension enregistré auprès de l'IANA>

X-TOKEN := <Préfixe "X-" ou "x-", suivi de n'importe quel jeton>

PARAMÈTRE := JETON "=" VALEUR

VALEUR := JETON / CHAÎNE ENTRE CITATIONS

JETON := 1*<tout caractère (US-ASCII) sauf ESPACE, CTL ou TSPECIALS>

SPÉCIALITÉS := "(" | ")" | "<" | ">" | "@" |

"," | ";" | ":" | "\" | <"> |

"/" | "[" | "]" | "?" | "="

Enregistrement IANA du type MIME

Le processus d'enregistrement du type de média MIME est décrit dans la RFC 2048. L'objectif du processus d'enregistrement est de faciliter l'enregistrement de nouveaux types de média, mais également de fournir une vérification de cohérence pour s'assurer que les nouveaux types sont bien pensés.

Arbres d'enregistrement

Les jetons de type MIME sont divisés en quatre classes, appelées « arbres d'enregistrement », chacune possédant ses propres règles d'enregistrement. Ces quatre arbres (IETF, fournisseur, personnel et expérimental) sont décrits dans le tableau D-2.

Tableau D-2. Quatre arborescences d'enregistrement de types de supports MIME

Exemple d'arbre d'enregistrement Description

Texte IETF/html (texte HTML) L'arbre IETF est destiné aux types qui ont une signification générale pour le

Communauté Internet. Les nouveaux types de supports d'arborescence IETF nécessitent l'approbation de l'Internet Engineering Steering Group (IESG) et une RFC de suivi des normes.

Les types d'arbres IETF n'ont pas de points (.) dans les jetons.

Fournisseur

(vnd.) image/vnd.fpx

(Image Kodak FlashPix) L'arborescence des fournisseurs est destinée aux types de supports utilisés par les produits disponibles dans le commerce. L'examen public des nouveaux types de fournisseurs est encouragé, mais non obligatoire.

Les types d’arborescence des fournisseurs commencent par « vnd. ».

Personnel/Vanité

(prs.) image/prs.btif

(format de gestion des chèques interne utilisé par la Nations Bank) Les supports privés, personnels ou personnalisés peuvent être enregistrés dans l'arborescence personnelle. Ces supports ne seront pas distribués commercialement.

Les types d’arbres personnels commencent par « prs. ».

Expérimental

(x- ou x.) application/x-tar (archive tar Unix). L'arborescence expérimentale est destinée aux types de supports non enregistrés ou expérimentaux. Comme il est relativement simple d'enregistrer un nouveau fournisseur ou un type de support personnel, les logiciels ne doivent pas être largement distribués à l'aide de types x.

Les types d’arbres expérimentaux commencent par « x. » ou « x- ».

Processus d'inscription

Lisez attentivement la RFC 2048 pour plus de détails sur l’enregistrement du type de média MIME.

Le processus d'enregistrement de base n'est pas un processus de normalisation formel ; il s'agit simplement d'une procédure administrative visant à vérifier la validité des nouveaux types auprès de la communauté et à les enregistrer dans un registre, sans délai. Le processus suit les étapes suivantes :

1. Présentez le type de média à la communauté pour examen.

Envoyez une proposition d'enregistrement de type de média à la liste de diffusion ietf-types@iana.org pour une période d'examen de deux semaines. Cette publication publique sollicite des commentaires sur le choix du nom, l'interopérabilité et les implications en matière de sécurité. Le préfixe « x- » spécifié dans la RFC 2045 peut être utilisé jusqu'à la finalisation de l'enregistrement.

Enregistrement IANA du type MIME

2. Approbation IESG (pour l'arbre IETF uniquement).

Si le type de média est enregistré dans l'arborescence IETF, il doit être soumis à l'IESG pour approbation et doit être accompagné d'une RFC de suivi des normes.

3. Enregistrement IANA.

Dès que le type de média répond aux exigences d'approbation, l'auteur peut soumettre la demande d'enregistrement à l'IANA, en utilisant le modèle de courriel de l'exemple D-1 et en envoyant les informations à

ietf-types@iana.org . L'IANA enregistrera le type de média et mettra la demande d'enregistrement à la disposition de la communauté à l'adresse http://www.isi.edu/in-notes/iana/assignments/media-types/.

Règles d'inscription

L'IANA enregistrera les types de médias dans l'arborescence de l'IETF uniquement en réponse à une communication de l'IESG indiquant qu'un enregistrement donné a été approuvé.

Les types de fournisseurs et de personnes seront enregistrés automatiquement par l'IANA et sans aucun examen formel, à condition que les conditions minimales suivantes soient remplies :

1. Les types de médias doivent fonctionner comme des formats de médias réels. Les types agissant comme des codages de transfert ou des jeux de caractères ne peuvent pas être enregistrés comme tels.

2. Tous les types de médias doivent avoir des noms de type et de sous-type appropriés. Tous les noms de type doivent être définis par des RFC normalisées. Tous les noms de sous-type doivent être uniques, conformes à la grammaire MIME et contenir les préfixes d'arborescence appropriés.

3. Les types d'arbres personnels doivent fournir une spécification de format ou un pointeur vers une spécification de format.

4. Les considérations de sécurité mentionnées ne doivent pas être manifestement fausses. Tous les développeurs de logiciels Internet doivent contribuer à prévenir les failles de sécurité.

Modèle d'inscription

L'enregistrement IANA se fait par courriel. Il suffit de remplir un formulaire d'inscription selon le modèle présenté dans l'exemple D-1 et de l'envoyer à ietf-types@iana.org* .

Exemple D-1. Modèle d'e-mail d'enregistrement MIME IANA

À : ietf-types@iana.org

Objet : Enregistrement du type de média MIME XXX/YYY Nom du type de média MIME :

* La structure légère du formulaire rend les informations soumises accessibles à l'utilisateur, mais difficiles à traiter par un ordinateur. C'est l'une des raisons pour lesquelles il est difficile de trouver un résumé lisible et bien organisé des types MIME, et c'est pourquoi nous avons créé les tableaux qui concluent cette annexe.

Exemple D-1. Modèle d'e-mail d'enregistrement MIME IANA (suite)
Nom du sous-type MIME :

Paramètres requis :

Paramètres optionnels :

Considérations relatives au codage :

Considérations de sécurité :

Considérations d’interopérabilité :

Spécification publiée :

Applications qui utilisent ce type de média :

Informations Complémentaires:

Nombre(s) magique(s) :

Extension(s) de fichier :

Code(s) de type de fichier Macintosh :

Personne et adresse email à contacter pour plus d'informations :

Utilisation prévue : (L'une des suivantes : COMMUN, USAGE LIMITÉ ou OBSOLÈTE) Auteur/Contrôleur des modifications : (Toute autre information que l'auteur juge intéressante peut être ajoutée sous cette ligne.)

Registre des types de médias MIME

Les formulaires soumis sont accessibles depuis le site web de l'IANA (http://www.iana.org). Au moment de la rédaction de ce document, la base de données des types de médias MIME est hébergée sur un serveur web de l'ISI, à l'adresse http://www.isi.edu/in-notes/iana/assignments/media-types/.

Les types de médias sont stockés dans une arborescence de répertoires, structurée par type principal et sous-type, avec un fichier feuille par type de média. Chaque fichier contient l'e-mail de soumission. Malheureusement, chaque personne remplit le formulaire d'inscription de manière légèrement différente, de sorte que la qualité et le format des informations varient selon les soumissions. (Dans les tableaux de cette annexe, nous avons tenté de combler les lacunes des inscrits.)

Tables de types MIME

Cette section résume des centaines de types MIME dans 10 tableaux. Chaque tableau répertorie les

Types de médias MIME au sein d'un type principal particulier (image, texte, etc.).

Les informations proviennent de nombreuses sources, notamment du registre des types de médias de l'IANA, du fichier mime.types d'Apache

et de diverses pages web. Nous avons passé plusieurs jours à affiner les données, à corriger les failles et à inclure des résumés descriptifs issus de références croisées afin de les rendre plus utiles.

Il s'agit probablement de la liste tabulaire des types MIME la plus détaillée jamais compilée. Nous espérons qu'elle vous sera utile !

application/*

Le tableau D-3 décrit de nombreux types de supports MIME spécifiques à l'application.

Tableau D-3. Types MIME « Application »

Type MIME Description Extension Contact et référence

application/activemessage Prend en charge le système de groupware Active Mail. « Active Mail : un cadre pour les applications de groupware intégrées » dans Lectures sur le groupware et

Coopérative assistée par ordinateur

Travail, Ronald M. Baecker, éd.,

Morgan Kaufmann, ISBN

1558602410

application/andrew-inset Prend en charge la création de contenu multimédia avec la boîte à outils Andrew. Développement d'applications multimédia faciles avec la boîte à outils Andrew,

Nathaniel S. Borenstein, Prentice Hall, ASIN 0130366331 nsb@bellcore.com

application/applefile : permet la transmission de données MIME avec des informations spécifiques à Apple/Macintosh, tout en autorisant l'accès général aux données utilisateur non spécifiques. RFC 1740

application/atomicmail. ATOMICMAIL était un projet de recherche expérimental de Bellcore, conçu pour inclure des programmes dans les messages électroniques, exécutés lors de leur lecture. ATOMICMAIL devient rapidement obsolète au profit de safe-tcl. Référence du langage ATOMICMAIL

Manuel », Nathaniel S. Borenstein,

Mémorandum technique de Bellcore

MC ARH-018429

application/batch-SMTP définit un type de contenu MIME adapté au tunneling d'une transaction de messagerie ESMTP via tout transport compatible MIME. RFC 2442

application/beep+xml prend en charge le protocole d'interaction BEEP. BEEP permet l'échange simultané et indépendant de messages MIME entre homologues, généralement au format texte structuré XML. RFC 3080

Type MIME Description Extension Contact et référence

application/cals-1840 prend en charge les échanges par e-mail MIME de données numériques du Département de la Défense des États-Unis précédemment échangées par tapem, conformément à la norme MIL-STD-1840. RFC 1895

Common Ground est un programme d'échange et de distribution de documents électroniques qui permet aux utilisateurs de créer des documents que chacun peut consulter, rechercher et imprimer, sans avoir besoin d'installer les applications ou les polices de création nécessaires. Nick Gault

Logiciel No Hands ngault@nohands.com

Application/CyberCash prend en charge le paiement par carte bancaire via le protocole CyberCash. Lorsqu'un utilisateur effectue un paiement, le commerçant envoie au client un message de type MIME « application/CyberCash ». RFC 1898

application/dca-rft Architecture de contenu de documents IBM. « Architecture de contenu de documents IBM/Référence de texte de formulaire révisable », document numéro SC230758-1, International Business

Machines

application/dec-dx Format de transfert de documents DEC. « Cahier technique sur la transmission de documents numériques (DX) », document numéro EJ29141-86, Digital

Société d'équipement

application/dvcs Prend en charge les protocoles utilisés par un Data

Serveur de validation et de certification (DVCS), qui agit comme tiers de confiance dans une infrastructure de sécurité à clé publique. RFC 3029

application/EDI-Consent Prend en charge les échanges bilatéraux via l'échange de données informatisées (EDI), en utilisant des spécifications non standard. http://www.isi.edu/in-notes/iana/assignations/media-types/ application/EDI-Consent

application/EDI-X12 Prend en charge les échanges bilatéraux via l'échange de données informatisé (EDI), en utilisant les spécifications EDI ASC X12. http://www.isi.edu/in-notes/iana/assignations/media-types/ application/EDI-X12

application/EDIFACT Prend en charge les échanges bilatéraux via l'échange de données informatisé (EDI), en utilisant les spécifications EDIFACT. http://www.isi.edu/in-notes/iana/assignations/media-types/application/EDIFACT

Application/e-boutique inconnue. Steve Katz

Architecture système Boutique steve_katz@eshop.com

application/font-tdpfr Définit une ressource de police portable (PFR) contenant un ensemble de formes de glyphes, chacune associée à un code de caractère. RFC 3073

Type MIME Description Extension Contact et référence

application/http Utilisé pour enfermer un pipeline d'un ou plusieurs messages de requête ou de réponse HTTP (non mélangés). RFC 2616

application/hyperstudio Prend en charge le transfert de fichiers hypermédias éducatifs HyperStudio. stk http://www.hyperstudio.com

application/iges Format couramment utilisé pour l'échange de modèles CAO. « ANS/US PRO/IPO-100 »

Association américaine des données sur les produits

2722 Merrilee Drive, bureau 200

Fairfax, Virginie 22031-4499

application/index application/index.cmd application/index.obj application/index.response application/index.vnd Prise en charge du protocole d'indexation commun (CIP). CIP est une évolution du service d'annuaire Whois++, utilisé pour transmettre les informations d'indexation d'un serveur à un autre afin de rediriger et de répliquer les requêtes via un système de base de données distribué. RFC 2652 et RFC 2651, 1913 et 1914

application/iotp : prend en charge les messages IOTP (Internet Open Trading Protocol) via HTTP. RFC 2935

application/ipp prend en charge le protocole d'impression Internet (IPP) sur HTTP. RFC 2910

application/mac-binhex40 Encode une chaîne d'octets de 8 bits en une chaîne d'octets de 7 bits, ce qui est plus sûr pour certaines applications (mais pas aussi sûr que l'encodage base-64 de 6 bits). hqx RFC 1341

application/mac-compactpro Depuis Apache mime.types. cpt

application/macwriteii Claris MacWrite II.

Les objets MARC sont des notices de catalogage lisibles par machine, des normes pour la représentation et la communication d'informations bibliographiques et connexes. mrc RFC 2220

application/mathematica application/mathematica-old Prend en charge Mathematica et Math-

Logiciel d'analyse numérique Reader. nb, ma,

mb Le livre Mathematica, Stephen

Wolfram, Université de Cambridge

Presse, ISBN 0521643147

application/msword Type MIME Microsoft Word. doc

application/news-message-id RFC 822 (ID de message), 1036

(application aux actualités), et 977

(NNTP)

Application/NewsTransmission : permet la transmission d'articles d'actualité par courriel ou autre moyen de transport. RFC 1036

application/ocsp-request Prend en charge le protocole OCSP (Online Certificate Status Protocol), qui permet de vérifier la validité d'un certificat numérique sans nécessiter de listes de révocation de certificats locales. orq RFC 2560

Type MIME Description Extension Contact et référence

application/ocsp-response Identique à ci-dessus. ors RFC 2560

application/octet-stream Données binaires non classifiées. bin,dms,

lha, lzh, exe, classe RFC 1341

application/oda Utilisé pour les informations codées selon les normes ODA (Office Document Architecture), en utilisant le format de représentation ODIF (Office Document Interchange Format). La ligne ContentType doit également spécifier un couple attribut/valeur indiquant le profil d'application du document (DAP), comme dans :

Type de contenu : application/oda ; profil = Q112 oda RFC 1341

ISO 8613 ; « Traitement de l'information : texte et système bureautique ; architecture des documents bureautiques (ODA) et

Format d'échange (ODIF) », partie

1-8, 1989

application/parityfec Codage de parité avec correction d'erreur directe pour les flux de données RTP. RFC 3009

Application/PDF Fichiers PDF Adobe. pdf Voir Manuel de référence du format de document portable, Adobe Systems, Inc., Addison Wesley, ISBN 0201626284

application/pgp-encrypted Données chiffrées PGP. RFC 2015

application/pgp-keys Blocs de clés publiques PGP. RFC 2015

application/pgp-signature Signature cryptographique PGP. RFC 2015

application/pkcs10 Système de chiffrement à clé publique n° 10 : le type de corps application/pkcs10 doit être utilisé pour transférer une demande de certification PKCS n° 10. p10 RFC 2311

application/pkcs7-mime Système de chiffrement à clé publique n° 7 : ce type est utilisé pour transporter des objets PKCS #7 de plusieurs types, notamment envelopedData et signedData. p7m RFC 2311

application/pkcs7-signature Système de chiffrement à clé publique n° 7 : ce type contient toujours un seul objet PKCS #7 de type signedData. p7s RFC 2311

application/pkix-cert Transporte les certificats X.509. cer RFC 2585

application/pkix-crl Transporte les listes de révocation de certificats X.509. crl RFC 2585

application/pkixcmp Format de message utilisé par les protocoles de gestion des certificats d'infrastructure à clés publiques X.509. pki RFC 2510

application/postscript Un fichier graphique Adobe PostScript (programme). ai, ps,

eps RFC 2046

application/prs.alvestrand.

Programme « TimeTracker » de Harald T. Alvestrand. http://domen.uninett.no/~hta/ titrax/

Type MIME Description Extension Contact et référence

application/prs.cww CU-Writer pour Windows. cw, cww Dr. Somchai Prasitjutrakul somchaip@chulkn.car.chula.ac.th

application/prs.nprend Inconnu. rnd, rct John M. Doggett jdoggett@tiac.net

Application/impression à distance : contient les métadonnées utilisées lors de l'impression à distance, pour la page de garde de l'imprimante. RFC 1486

Marshall T. Rose mrose@dbc.mtview.ca.us

application/riscos Binaires Acorn RISC OS. Référence du programmeur RISC OS

Manuels, Acorn Computers, Ltd.,

ISBN 1852501103

Application/SDP : SDP est destiné à décrire des sessions multimédias en direct pour l'annonce, l'invitation et d'autres formes d'initiation de sessions multimédias. RFC 2327

Henning Schulzrinne hgs@cs.columbia.edu

application/set-payment application/set-paymentinitiation

application/set-registration application/set-registrationinitiation Prend en charge le protocole de paiement de transaction électronique sécurisé SET. http://www.visa.com http://www.mastercard.com

application/sgml-opencatalog Destiné à être utilisé avec les systèmes qui prennent en charge la spécification SGML Open TR9401:1995 « Gestion des entités ». SGML Open

910 Beaver Grade Road, #3008 Coraopolis, PA 15109 info@sgmlopen.org

application/sieve Script de filtrage de courrier Sieve. RFC 3028

application/slate Le format de document BBN/Slate est publié dans le cadre de l'ensemble de documentation standard distribué avec le produit BBN/Slate. Gestionnaire de produits BBN/Slate

Systèmes et technologies BBN

10, rue Moulton

Cambridge, MA 02138

application/smil Le langage d'intégration multimédia synchronisé (SMIL) intègre un ensemble d'objets multimédias indépendants dans une présentation multimédia synchronisée. smi, smil http://www.w3.org/AudioVideo/

application/tve-trigger Prend en charge les URL intégrées dans les récepteurs de télévision améliorés. « SMPTE : Essence de données déclaratives,

Contenu de niveau 1 », produit par le

Société des ingénieurs du cinéma et de la télévision http://www.smpte.org

application/vemmi Norme vidéotex améliorée. RFC 2122

application/vnd.3M.Post-itNotes Utilisé par le contrôle/plug-in Internet « Post-it® Notes for Internet Designers ». pwn http://www.3M.com/psnotes/

Type MIME Description Extension Contact et référence

application/vnd.accpac.

simply.aso Simple Comptable v7.0 et supérieur.

Les fichiers de ce type sont conformes aux spécifications Open Financial Exchange v1.02. aso http://www.ofx.net

application/vnd.accpac.

simply.imp Utilisé par Simply Accounting v7.0 et supérieur, pour importer ses propres données. imp http://www.ofx.net

application/vnd.acucobol Exécution ACUCOBOL-GT. Dovid Lubin dovid@acucobol.com

application/vnd.aether.imp Prend en charge les communications par messagerie instantanée à faible consommation de temps d'antenne entre un service de messagerie instantanée, tel qu'AOL Instant Messenger, Yahoo! Messenger ou MSN Messenger, et un

Ensemble de logiciels clients de messagerie instantanée sur un appareil sans fil. Spécification du protocole de messagerie instantanée sans fil (IMP) disponible sous licence auprès d'Aether Systems.

application/vnd.anser-webcertificate-issue-initiation Déclencheur permettant aux navigateurs Web de lancer le client de terminal ANSER-WEB. cii Hiroyoshi Mori mori@mm.rd.nttdata.co.jp

application/vnd.anser-webfunds-transfer-initiation Identique à ci-dessus. fti Identique à ci-dessus

application/vnd.audiograph AudioGraph. aep Horia Cristian

HCSlusanschi@massey.ac.nz

application/vnd.bmi Format graphique BMI par CADAM Systems. bmi Tadashi Gotoh

tgotoh@cadamsystems.co.jp

application/vnd. businessobjects BusinessObjects 4.0 et versions ultérieures. rep

application/vnd.canon-cpdl application/vnd.canon-lips Prend en charge les produits d'imagerie de bureau Canon, Inc. Shin Muto

shinmuto@pure.cpdc.canon.co.jp

application/vnd.claymore Claymore.exe. cla Ray Simpson ray@cnation.com

application/vnd.commercebattelle Prend en charge un mécanisme générique de délimitation des informations basées sur les cartes à puce, pour le commerce numérique, l'identification, l'authentification et l'échange d'informations sur les titulaires de cartes à puce. ica, icf, icd, icc, ic0, ic1, ic2, ic3, ic4, ic5, ic6, ic7, ic8 David C. Applebaum applebau@131.167.52.15

application/vnd. commonspace Permet une transmission correcte des

Documents CommonSpace™ via

Processus basés sur MIME. Common-

Space est publié par Sixth Floor Media, une filiale de Houghton-Mifflin Company. csp, cst Ravinder Chandhok chandhok@within.com

application/vnd.contact.cmsg Utilisé pour la BASE DE DONNÉES CIM du logiciel CONTACT. cdbcmsg Frank Patz fp@contact.de http://www.contact.de

Type MIME Description Extension Contact et référence

application/vnd.cosmocaller Permet de télécharger des fichiers contenant des paramètres de connexion à partir de sites Web, appelle l'application CosmoCaller pour interpréter les paramètres et initie des connexions avec le serveur CosmoCallACD. cmc Steve Dellutri sdellutri@cosmocom.com

application/vnd.ctc-posml PosML de Continuum Technology. pml Bayard Kohlhepp bayardk@ctcexchange.com

application/vnd.cupspostscript

application/vnd.cups-raster application/vnd.cups-raw Prend en charge les serveurs et clients Common UNIX Printing System (CUPS). http://www.cups.org

application/vnd.cybank : type de données propriétaire pour les données Cybank. Nor Helmee B. Abd. Halim helmee@cybank.net http://www.cybank.net

application/vnd.dna DNA est conçu pour permettre l'accès Web facile à n'importe quelle application Windows 32 bits. dna Meredith Searcy msearcy@newmoon.com

application/vnd.dpgraph Utilisé par DPGraph 2000 et MathWare Cyclone. dpg, mwc, dpgraph David Parker

http://www.davidparker.com

application/vnd.dxr Rapports Digital Xpress par PSI Technologies. dxr Michael Duffy miked@psiaustin.com

application/vnd.ecdis-update Prend en charge les applications ECDIS. http://www.sevencs.com

application/vnd.ecowin.graphique application/vnd.ecowin.

demande de fichier application/vnd.ecowin.

mise à jour du fichier

application/vnd.ecowin.série application/vnd.ecowin.

demande de série application/vnd.ecowin.

mise à jour de la série EcoWin. mag Thomas Olsson thomas@vinga.se

application/vnd.enliven Prend en charge la diffusion de contenu multimédia interactif Enliven. nml Paul Santinelli

psantinelli@narrative.com

application/vnd.epson.esf Contenu propriétaire pour Seiko Epson QUASS Stream Player. esf Shoji Hoshina

Hoshina.Shoji@exc.epson.co.jp

application/vnd.epson.msf Contenu propriétaire pour Seiko Epson QUASS Stream Player. msf Identique à ci-dessus

application/vnd.epson.

Contenu propriétaire de QuickAnime Player pour Seiko Epson QuickAnime. qam Yu Gu

guyu@rd.oda.epson.co.jp

application/vnd.epson.salt Contenu exclusif pour Seiko Epson SimpleAnimeLite Player. slt Yasuhito Nagatomo naga@rd.oda.epson.co.jp

application/vnd.epson.ssf Contenu propriétaire pour Seiko Epson QUASS Stream Player. ssf Shoji Hoshina

Hoshina.Shoji@exc.epson.co.jp

Type MIME Description Extension Contact et référence

application/vnd.ericsson.

Quickcall Doubleur téléphonique Quick Call. qcall, qca Paul Tidwell

paul.tidwell@ericsson.com http://www.ericsson.com

application/vnd.eudora.data Eudora version 4.3 et ultérieures. Pete Resnick , presnick@qualcomm.com

application/vnd.fdf Format de données Adobe Forms. « Format de données de formulaires », technique

Note 5173, Adobe Systems

application/vnd.ffsns Utilisé pour la communication de l'application avec Smart Delivery de FirstFloor. Mary Holstege holstege@firstfloor.com

application/vnd.FloGraphIt NpGraphIt.gph

application/vnd.framemaker Fichiers Adobe FrameMaker. fm, mif, livre http://www.adobe.com

application/vnd.fsc.

weblaunch prend en charge le logiciel de simulation de golf de Friendly Software Corporation. fsc Derek Smith

derek@friendlysoftware.com

application/vnd.fujitsu.oasys application/vnd.fujitsu.oasys2 Prend en charge le logiciel OASYS de Fujitsu. oas Nobukazu Togashi

togashi@ai.cs.fujitsu.co.jp

application/vnd.fujitsu.oasys2 Prend en charge le logiciel OASYS V2 de Fujitsu. oa2 Identique à ci-dessus

application/vnd.fujitsu.oasys3 Prend en charge le logiciel OASYS V5 de Fujitsu. oa3 Seiji Okudaira

okudaira@candy.paso.fujitsu.co.jp

application/vnd.fujitsu.

oasysgp prend en charge le logiciel OASYS GraphPro de Fujitsu. fg5 Masahiko Sugimoto sugimoto@sz.sel.fujitsu.co.jp

application/vnd.fujitsu.

oasysprs prend en charge le logiciel de présentation OASYS de Fujitsu. bh2 Masumi Ogita

ogita@oa.tfl.fujitsu.co.jp

application/vnd.fujixerox.ddd Prend en charge EDMICS 2000 et DocuFile de Fuji Xerox. ddd Masanori Onda

Masanori.Onda@fujixerox.co.jp

application/vnd.fujixerox.

docuworks prend en charge les logiciels DocuWorks Desk et DocuWorks Viewer de Fuji Xerox. xdw Yasuo Taguchi

yasuo.taguchi@fujixerox.co.jp

application/vnd.fujixerox.

docuworks.binder Prend en charge les logiciels DocuWorks Desk et DocuWorks Viewer de Fuji Xerox. xbd Identique à ci-dessus.

application/vnd.fut-misnet Inconnu. Jaan Pruulmann jaan@fut.ee

application/vnd.grafeq Permet aux utilisateurs de GrafEq d'échanger des documents GrafEq via le Web et par courrier électronique. gqf, gqs http://www.peda.com

application/vnd.grooveaccount Groove est un système de communication peer-to-peer mettant en œuvre un espace virtuel pour l'interaction en petits groupes. gac Todd Joseph

todd_joseph@groove.net

application/vnd.grooveidentity-message Identique à ci-dessus. gim Identique à ci-dessus

application/vnd.grooveinjector Identique à ci-dessus. grv Identique à ci-dessus

application/vnd.groove-toolmessage Identique à ci-dessus. gtm Identique à ci-dessus

Type MIME Description Extension Contact et référence

application/vnd.groove-tooltemplate Identique à ci-dessus. tpl Identique à ci-dessus

application/vnd.groove-vcard Identique à ci-dessus. vcg Identique à ci-dessus

application/vnd.hhe.lessonplayer Prend en charge les logiciels LessonPlayer et PresentationEditor. les Randy Jones Harcourt E-Learning randy_jones@archipelago.com

fichier application/vnd.hp-HPGL HPGL. Références HP-GL/2 et HP RTL.

Guide, Addison Wesley, ISBN

0201310147

application/vnd.hp-hpid Prend en charge le logiciel Instant Delivery de Hewlett-Packard. hpi, hpid http://www.instant-delivery.com

application/vnd.hp-hps Prend en charge le logiciel WebPrintSmart de Hewlett-Packard. hps http://www.hp.com/go/ webprintsmart_mimetype_specs/

application/vnd.hp-PCL application/vnd.hp-PCLXL Fichiers d'imprimante PCL. pcl « Package de documentation du manuel de référence technique PCL-PJL », référence HP 5012-0330

application/vnd.httphone Système de voix sur IP asynchrone HTTPhone. Franck LeFevre franck@k1info.com

application/vnd.hzn-3dcrossword Utilisé pour encoder les mots croisés d'Horizon, un aperçu de demain. x3d James Minnis

james_minnis@glimpse-oftomorrow.com

application/vnd.ibm.

Services d'impression afplinedata (PSF), fonction de conversion et d'indexation AFP (ACIF). Roger Buis, buis@us.ibm.com

application/vnd.ibm.MiniPay Logiciel d'authentification et de paiement MiniPay. mpy Amir Herzberg amirh@vnet.ibm.com

application/vnd.ibm.modcap Contenu de document d'objet mixte. list3820, listafp, afp,

pseg3820 Reinhard Hohensee rhohensee@vnet.ibm.com

« Référence d'architecture de contenu de document d'objet mixte », publication IBM SC31-6802

application/vnd.informixvisionary Informix Visionary. vis Christopher Gales

christopher.gales@informix.com

application/vnd.intercon.

Formnet prend en charge le logiciel FormNet d'Intercon Associates. xpw, xpx Thomas A. Gurak

assoc@intercon.roc.servtech.com

application/vnd.intertrust.

application digibox/vnd.intertrust.

nncp prend en charge l'architecture InterTrust pour un commerce électronique sécurisé et la gestion des droits numériques. InterTrust Technologies

460 Oakmead Parkway Sunnyvale, CA 94086 États-Unis info@intertrust.com http://www.intertrust.com

application/vnd.intu.qbo Destiné à être utilisé uniquement avec QuickBooks 6.0 (Canada). qbo Greg Scratchley

greg_scratchley@intuit.com

Le format de ces fichiers est décrit dans les spécifications de l'Open Financial Exchange, disponibles sur http://www.ofx.net

Type MIME Description Extension Contact et référence

application/vnd.intu.qfx Destiné à être utilisé uniquement avec Quicken 99 et les versions suivantes. qfx Identique à ci-dessus

application/vnd.is-xpr Express par Infoseek. xpr Satish Natarajan satish@infoseek.com

application/vnd.japannetdirectory-service
application/vnd.japannetjpnstore-wakeup
application/vnd.japannetpayment-wakeup

application/vnd.japannetregistration

application/vnd.japannetregistration-wakeup
application/vnd.japannetsetstore-wakeup
application/vnd.japannetverification

application/vnd.japannetverification-wakeup Prend en charge les logiciels de sécurité, d'authentification et de paiement JapanNet de Mitsubishi Electric. Jun Yoshitake

yositake@iss.isl.melco.co.jp

application/vnd.koan Prend en charge la lecture automatique des fichiers musicaux Koan sur Internet, par des applications d'assistance telles que SSEYO Koan Netscape Plugin. skp, skd, skm, skt Peter Cole

pcole@sseyod.demon.co.uk

application/vnd.lotus-1-2-3 Lotus 1-2-3 et Lotus approach. 123, sem. 1, sem. 3, sem. 4 Paul Wattenberger

Paul_Wattenberger@lotus.com

application/vnd.lotusapproach Lotus Approach. apr, vew Identique à ci-dessus

application/vnd.lotusfreelance Lotus Freelance. prz, pre Identique à ci-dessus

application/vnd.lotus-notes Lotus Notes. nsf, ntf, ndl, ns4, ns3, ns2, nsh, nsg Michael Laramie laramiem@btv.ibm.com

application/vnd.lotusorganizer Lotus Organizer. or3, or2, org Paul Wattenberger

Paul_Wattenberger@lotus.com

application/vnd.lotusscreencam Lotus ScreenCam. scm Identique à ci-dessus

application/vnd.lotuswordpro Lotus Word Pro. lwp, sam Identique à ci-dessus

application/vnd.mcd Logiciel de CAO Micro CADAM. mcd Tadashi Gotoh

tgotoh@cadamsystems.co.jp http://www.cadamsystems.co.jp

Type MIME Description Extension Contact et référence

application/vnd.mediastation.cdkey Prend en charge le protocole de communication CDROM à distance CDKey de Media Station. cdkey Henry Flurry

henryf@mediastation.com

application/vnd.meridianslingshot Slingshot de Meridian Data. Eric Wedel

Meridian Data, Inc.

5615 Scotts Valley Drive

Scotts Valley, Californie 95066

ewedel@meridian-data.com

application/vnd.mif Format d'échange FrameMaker. mif ftp://ftp.frame.com/pub/techsup/ techinfo/dos/mif4.zip

Mike Wexler

Adobe Systems, Inc. 333 W. San Carlos St. San Jose, CA 95110 États-Unis mwexler@adobe.com

application/vnd.minisofthp3000-save Format de sauvegarde NetMail 3000. Minisoft, Inc.

support@minisoft.com ftp://ftp.3k.com/DOC/ms92-saveformat.txt

application/vnd.mitsubishi.

Misty-guard.trustweb prend en charge le logiciel Trustweb de Mitsubishi Electric. Manabu Tanaka

mtana@iss.isl.melco.co.jp

application/vnd.Mobius.DAF Prend en charge le logiciel Mobius Management Systems. daf Celso Rodriguez crodrigu@mobius.com

Greg Chrzczon gchrzczo@mobius.com

application/vnd.Mobius.DIS Identique à ci-dessus. dis Identique à ci-dessus

application/vnd.Mobius.MBK Identique à ci-dessus. mbk Identique à ci-dessus

application/vnd.Mobius.MQY Identique à ci-dessus. mqy Identique à ci-dessus

application/vnd.Mobius.MSL Identique à ci-dessus. msl Identique à ci-dessus

application/vnd.Mobius.PLC Identique à ci-dessus. plc Identique à ci-dessus

application/vnd.Mobius.TXF Identique à ci-dessus. txf Identique à ci-dessus

application/vnd.motorola.

Flexsuite™ est un ensemble de protocoles de messagerie sans fil. Ce type de protocole est utilisé par les passerelles réseau des fournisseurs de services de messagerie sans fil, ainsi que par les systèmes d'exploitation et applications sans fil. Mark Patton

Groupe Réseaux Personnels Motorola fmp014@email.mot.com

Spécification FLEXsuite™ disponible auprès de Motorola sous contrat de licence approprié

application/vnd.motorola.

flexsuite.adsi FLEXsuite™ est un ensemble de protocoles de messagerie sans fil. Ce type de protocole offre un format compatible avec le sans fil permettant diverses solutions de chiffrement des données. Identique à ci-dessus.

Type MIME Description Extension Contact et référence

application/vnd.motorola.

flexsuite.fis FLEXsuite™ est un ensemble de protocoles de messagerie sans fil. Ce format, compatible avec les réseaux sans fil, permet de transmettre efficacement des informations structurées (actualités, bourse, météo, etc.) à un appareil sans fil. Voir ci-dessus.

application/vnd.motorola.

flexsuite.gotap FLEXsuite™ est un ensemble de protocoles de messagerie sans fil. Ce type fournit un format commun, compatible avec le sans fil, pour la programmation des attributs des appareils sans fil via des messages OTA. Identique à ci-dessus.

application/vnd.motorola.

flexsuite.kmr FLEXsuite™ est un ensemble de protocoles de messagerie sans fil. Ce type de protocole offre un format compatible avec le sans fil pour la gestion des clés de chiffrement. Identique à ci-dessus.

application/vnd.motorola.

flexsuite.ttc FLEXsuite™ est un ensemble de protocoles de messagerie sans fil. Ce type prend en charge un format compatible sans fil pour

une transmission efficace de texte grâce à la compression de texte par jeton. Identique à ci-dessus.

application/vnd.motorola.

flexsuite.wem FLEXsuite™ est un ensemble de protocoles de messagerie sans fil. Ce type de protocole offre un format compatible avec le sans fil pour la communication de courriels Internet vers des appareils sans fil. Identique à ci-dessus.

application/vnd.mozilla.

xul+xml Prend en charge la suite d'applications Internet Mozilla. xul Dan Rosen2 dr@netscape.com

application/vnd.ms-artgalry Prend en charge la galerie d'art de Microsoft. cil deansl@microsoft.com

application/vnd.ms-asf. ASF est un format de fichier multimédia dont le contenu est conçu pour être diffusé sur un réseau afin de prendre en charge les applications multimédias distribuées. Le contenu ASF peut inclure toute combinaison de supports (par exemple, audio, vidéo, images, URL, contenu HTML, MIDI, modélisation 2D et 3D, scripts et objets de divers types). asf Eric Fleischman ericf@microsoft.com http://www.microsoft.com/mind/0997/netshow/netshow.asp

application/vnd.ms-excel Feuille de calcul Microsoft Excel. xls Sukvinder S. Gill sukvg@microsoft.com

application/vnd.ms-lrm Propriété de Microsoft. lrm Eric Ledoux ericle@microsoft.com

application/vnd.mspowerpoint Présentation Microsoft PowerPoint. ppt Sukvinder S. Gill sukvg@microsoft.com

application/vnd.ms-project Fichier Microsoft Project. mpp Identique à ci-dessus

Type MIME Description Extension Contact et référence

application/vnd.ms-tnef : identifie une pièce jointe qui, en général, ne serait traitable que par une application compatible MAPI. Ce type est un format encapsulé de propriétés MAPI riches, telles que du texte enrichi et des icônes, qui pourraient autrement être dégradées par le transport de messagerie. Identique à ci-dessus.

application/vnd.ms-works Logiciel Microsoft Works. Identique à ci-dessus.

application/vnd.mseq MSEQ est un format multimédia compact adapté aux appareils sans fil. mseq Gwenael Le Bodic

Gwenael.le_bodic@alcatel.fr http://www.3gpp.org

application/vnd.msign Utilisé par les applications implémentant le protocole msign, qui demande des signatures aux appareils mobiles. Malte Borcherding

Malte.Borcherding@brokat.com

application/vnd.music-niff : fichiers musicaux NIFF. Cindy Grande

72723.1272@compuserve.com ftp://blackbox.cartah.washington.

edu/pub/NIFF/NIFF6A.TXT

application/vnd.musician MUSICIANscoringlanguage/encoding conçu et développé par RenaiScience Corporation. mus Robert G. Adams gadams@renaiscience.com

application/vnd.netfpx Destiné à la récupération dynamique d'informations d'images multirésolutions, telles qu'utilisées par Hewlett-Packard Company Imaging for Internet. fpx Andy Mutz andy_mutz@hp.com

application/vnd.noblenetdirectory Prend en charge le logiciel NobleNet Directory, acheté par RogueWave. nnd http://www.noblenet.com

application/vnd.noblenetsealer Prend en charge le logiciel NobleNet Sealer, acheté par RogueWave. nns http://www.noblenet.com

application/vnd.noblenet-

web Prend en charge le logiciel NobleNet Web, acheté par RogueWave. nnw http://www.noblenet.com

application/vnd.novadigm.

EDM prend en charge les produits RADIA et EDM de Novadigm. edm Phil Burgard

pburgard@novadigm.com

application/vnd.novadigm. EDX Identique à ci-dessus. edx Identique à ci-dessus

application/vnd.novadigm. EXT Identique à ci-dessus. ext Identique à ci-dessus

application/vnd.osa.

netdeploy prend en charge le logiciel de déploiement d'applications netDeploy d'Open Software Associates. ndc Steve Klos stevek@osa.com http://www.osa.com

application/vnd.palm Utilisé par les logiciels et applications système PalmOS, ce nouveau type, « application/vnd.palm », remplace l'ancien type « application/x-pilot ». prc, pdb, pqa, oprc Gavin Peacock gpeacock@palm.com

Type MIME Description Extension Contact et référence

application/vnd.pg.format Système de rapport standard propriétaire de Proctor & Gamble. str April Gandert

TN152

Chemin Procter & Gamble

Cincinnati, Ohio 45202

(513) 983-4249

application/vnd.pg.osasli Système de rapport standard propriétaire de Proctor & Gamble. ei6 Identique à ci-dessus

application/vnd. powerbuilder6 application/vnd. powerbuilder6-s application/vnd. powerbuilder7 application/vnd. powerbuilder7-s application/vnd. powerbuilder75 application/vnd. powerbuilder75-s Utilisé uniquement par les environnements d'exécution Sybase PowerBuilder versions 6, 7 et 7.5, non sécurisés et sécurisés. pbd Reed Shilts

reed.shilts@sybase.com

application/vnd.

previewsystems.box Aperçu du produit ZipLock/VBox de Preview Systems. box, vbox Roman Smolgovsky romans@previewsystems.com http://www.previewsystems.com

application/vnd.publisharedelta-tree Utilisé par Capella Computers

Environnement d'exécution PubliShare. qps Oren Ben-Kiki

publishare-delta-tree@capella.co.il

application/vnd.rapid Applications packagées rapides d'Emultek. zrp Itay Szekely etay@emultek.co.il

application/vnd.s3sms : intègre les mécanismes de transfert des produits Sonera SmartTrust à l'infrastructure Internet. Lauri Tarkkala

Lauri.Tarkkala@sonera.com http://www.smarttrust.com

application/vnd.seemail prend en charge la transmission de fichiers SeeMail. SeeMail est une application qui capture la vidéo et le son et utilise la compression bit à bit pour compresser et archiver les deux éléments en un seul fichier. Voir Steven Webb steve@wynde.com http://www.realmediainc.com

application/vnd.shana.

Formats de données des formulaires électroniques Shana. ifm Guy Selzler Shana Corporation gselzler@shana.com

application/vnd.shana.informed.formtemp Formats de données des formulaires électroniques Shana.itp Identique à ci-dessus

application/vnd.shana.informed.interchange Formats de données des formulaires électroniques Shana. iif, iif1 Identique à ci-dessus

application/vnd.shana.

formats de données des formulaires électroniques Shana informed.package. ipk, ipkg Identique à ci-dessus

application/vnd.street-stream, propriété de Street Technologies. Glenn Levitt Street Technologies streetd1@ix.netcom.com

Type MIME Description Extension Contact et référence

application/vnd.svd : fichiers SVD de Dateware Electronics. Scott Becker

dataware@compumedia.com

application/vnd.swiftview-ics Prend en charge SwiftView®. Randy Prakken tech@ndg.com

http://www.ndg.com/svm.htm

application/vnd.triscape.mxs Prend en charge Triscape Map Explorer. mxs Steven Simonoff scs@triscape.com

application/vnd.trueapp Fichiers True BASIC. tra J. Scott Hepler scott@truebasic.com

application/vnd.truedoc Propriété de Bitstream, Inc. Brad Chase

brad_chase@bitstream.com

application/vnd.ufdl Fichiers UFDL de l'UWI. ufdl, ufd, de Dave Manning dmanning@uwi.com http://www.uwi.com/

application/vnd.uplanet.alerte application/vnd.uplanet.

alerte-wbxml application/vnd.uplanet.

application bearer-choi-wbxml/vnd.uplanet.

application de choix du porteur/vnd.uplanet.

cacheop application/vnd.uplanet. cacheop-wbxml application/vnd.uplanet.

application de canal/vnd.uplanet.channel-wbxml

application/vnd.uplanet.list

application/vnd.uplanet.list-

application wbxml/vnd.uplanet.

application listcmd/vnd.uplanet.

listecmd-wbxml application/vnd.uplanet.

Formats de signal utilisés par le micro-navigateur UP Browser Unwired Planet (maintenant Openwave) pour appareils mobiles. iana-registrar@uplanet.com http://www.openwave.com

application/vnd.vcx Catalogue virtuel. vcx Taisuke Sugimoto

sugimototi@noanet.nttdata.co.jp

application/vnd.vectorworks Fichiers graphiques VectorWorks. mcd Paul C. Pharr

pharr@diehlgraphsoft.com

application/vnd.vidsoft.

Format de vidéoconférence VidConference. vsc Robert Hess hess@vidsoft.de

application/vnd.visio Fichiers Visio. vsd, vst, vsw, vss Troy Sandal troys@visio.com

Type MIME Description Extension Contact et référence

application/vnd.vividence.

Fichiers scriptfile Vividence. vsf, vtd,

vd Mark Risher markr@vividence.com

application/vnd.wap.sic Format d'indication de service WAP. sic,

wbxml WAP Forum Ltd http://www.wapforum.org

format de chargement du service WAP application/vnd.wap.slc.

Tout ce qui est conforme au Service

Spécification de chargement, disponible sur http://www.wapforum.org.slc,

wbxml Identique à ci-dessus

application/vnd.wap.wbxml Format XML binaire WAP WBXML pour les appareils sans fil. wbxml Identique à ci-dessus

« Contenu XML binaire WAP

Format — WBXML version 1.1”

application/vnd.wap.wmlc Format WAP WML pour les appareils sans fil. wmlc, wbxml Identique à ci-dessus

application/vnd.wap.

wmlscriptc Format WAP WMLScript. wmlsc Identique à ci-dessus

application/vnd.webturbo Format WebTurbo. wtb Yaser Rehem

Sapient Corporation yrehem@sapient.com

application/vnd.wrq-hp3000labelled Prend en charge les formats HP3000. support@wrq.com support@3k.com

application/vnd.wt.stf Prend en charge le logiciel Worldtalk. stf Bill Wohler wohler@worldtalk.com

application/vnd.xara Les fichiers Xara sont enregistrés par CorelXARA, un logiciel de graphisme vectoriel orienté objet écrit par Xara Limited (et commercialisé par Corel). xar David Matthewman david@xara.com http://www.xara.com

application/vnd.xfdl Fichiers XFDL de l'UWI. xfdl, xfd,

de Dave Manning dmanning@uwi.com http://www.uwi.com

application/vnd.yellowrivercustom-menu Prend en charge le plug-in Yellow River CustomMenu, qui fournit des menus déroulants de navigateur personnalisés. cmp yellowriversw@yahoo.com

application/whoispp-query : définit les requêtes du protocole Whois++ dans MIME. RFC 2957

application/whoisppresponse : définit les réponses du protocole Whois++ dans MIME. RFC 2958

Application/Wita Wang Information Transfer Architecture. Document n° 715-0050A, Laboratoires Wang, campbell@redsox.bsw.com

application/wordperfect5.1 Documents WordPerfect.

application/x400-bp Transporte toute partie du corps X.400 pour laquelle il n'existe pas de mappage IANA enregistré. RFC 1494

Type MIME Description Extension Contact et référence

application/x-bcpio Archives CPIO binaires à l'ancienne. bcpio

application/x-cdlink Permet l'intégration de supports CD-ROM dans des pages Web. vcd http://www.cdlink.com

application/x-chess-pgm Depuis Apache mime.types. pgn

application/x-compress Données binaires d'Unix compress. z

fichier d'archive CPIO application/x-cpio. cpio

application/x-csh Scripts CSH. csh

application/x-director Fichiers Macromedia Director. dcr, dir, dxr

application/x-dvi Fichiers TeX DVI. dvi

application/x-futuresplash Depuis Apache mime.types. spl

application/x-gtar Archives tar GNU. gtar

application/x-gzip Données compressées GZIP. gz

application/x-hdf Depuis Apache mime.types. hdf

fichiers JavaScript application/x-javascript. js

application/x-koan Prend en charge la lecture automatique des fichiers musicaux Koan sur Internet, par des applications d'assistance telles que SSEYO Koan Netscape Plugin. skp, skd, skt, skm

application/x-latex Fichiers LaTeX. latex

application/x-netcdf Fichiers NETCDF. nc, cdf

Scripts SH application/x-sh. sh

application/x-shar Archives SHAR. shar

application/x-shockwaveflash Fichiers Macromedia Flash. swf

application/x-stuffit Archives StuffIt. sit

Archives CPIO d'application/x-sv4cpio Unix SysV R4. sv4cpio

application/x-sv4crc Archives CPIO Unix SysV R4 avec CRC. sv4crc

archives TAR application/x-tar. tar

Scripts TCL application/x-tcl. tcl

fichiers TeX application/x-tex. tex

application/x-texinfo Fichiers d'informations TeX. texinfo, texi

application/x-troff Fichiers TROFF. t, tr, roff

application/x-troff-man Pages de manuel TROFF Unix. man

application/x-troff-me Fichiers TROFF+me. moi

application/x-troff-ms Fichiers TROFF+ms ms

Type MIME Description Extension Contact et référence

application/x-ustar Le format d'échange tar étendu. ustar Voir les spécifications IEEE 1003.1(1990)

application/x-wais-source Structure source WAIS. src

application/xml Fichier au format Extensible Markup Language (utilisez text/xml si vous souhaitez que le fichier soit traité comme du texte brut par les navigateurs, etc.). xml, dtd RFC 2376

application/zip Archives zip PKWARE. zip

audio/*

Le tableau D-4 résume les types de contenu audio.

Tableau D-4. Types MIME « audio »

Type MIME Description Extension Contact et référence

Codage audio ADPCM 8 kHz audio/32 kA/dpcm. RFC 2421

audio/basic Audio codé avec un signal PCM RNIS u-law monophonique 8 kHz 8 bits. au, snd RFC 1341

Audio/G.772.1 : Le G.722.1 compresse les signaux audio de 50 Hz à 7 kHz en 24 kbit/s ou 32 kbit/s. Il peut être utilisé pour la parole, la musique et d'autres types de son. RFC 3047

Audio/L16 est basé sur L16, décrit dans la RFC 1890. L16 désigne des données audio non compressées, utilisant une représentation signée sur 16 bits. RFC 2586

audio/MP4A-LATM Audio MPEG-4. RFC 3016

Fichiers musicaux MIDI audio/midi. mid, midi, kar

audio/mpeg Fichiers audio codés MPEG. mpga, mp2, mp3 RFC 3003

audio/parityfec Correction d'erreurs directe basée sur la parité pour l'audio RTP. RFC 3009

audio/prs.sid Fichiers audio Commodore 64 SID. sid, psid http://www.geocities.com/ SiliconValley/Lakes/5147/sidplay/ docs.html#fileformats

événement audio/téléphonique Événement téléphonique logique. RFC 2833

Modèle de son téléphonique audio/tonalité. RFC 2833

audio/vnd.cns.anp1 prend en charge les fonctionnalités vocales et de messagerie unifiée disponibles sur la plateforme de services réseau Access NP de Comverse Network Systems. Ann McLaughlin

Systèmes de réseau Comverse amclaughlin@comversens.com

Tableau D-4. Types MIME « audio » (suite)

Type MIME Description Extension Contact et référence

audio/vnd.cns.inf1 prend en charge les fonctionnalités vocales et de messagerie unifiée disponibles sur la plateforme de services réseau TRILOGUE Infinity de Comverse Network Systems. Identique à ci-dessus.

audio/vnd.digital-winds La musique de Digital Winds est une musique MIDI infinie, reproductible et interactive dans de très petits packages (<3K). eol Armands Strazds

armands.strazds@medienhausbremen.de

audio/vnd.everad.plj Encodage audio propriétaire EverAD. plj Tomer Weisberg tomer@everad.com

audio/vnd.lucent.voice Messagerie vocale incluant le système de messagerie multimédia Intuity™ AUDIX® de Lucent Technologies et le lecteur vocal Lucent. lvp Frederick Block rickblock@lucent.com http://www.lucent.com/lvp/

audio/vnd.nortel.vbk Codage audio propriétaire de Nortel Networks Voice Block. vbk Glenn Parsons

Glenn.Parsons@NortelNetworks.com

audio/vnd.nuera.ecelp4800 Codage audio et vocal propriétaire de Nuera Communications, disponible dans les passerelles de voix sur IP, les terminaux, les serveurs d'applications Nuera et en tant que service multimédia pour diverses plates-formes hôtes et systèmes d'exploitation. ecelp4800 Michael Fox mfox@nuera.com

audio/vnd.nuera.ecelp7470 Identique à ci-dessus. ecelp7470 Identique à ci-dessus

audio/vnd.nuera.ecelp9600 Identique à ci-dessus. ecelp9600 Identique à ci-dessus

audio/vnd.octel.sbc Encodage à débit variable de 18 kbit/s en moyenne, utilisé pour la messagerie vocale sur les plateformes Sierra™, Overture™ et IMA™ de Lucent Technologies. Jeff Bouis jbouis@lucent.com

audio/vnd.qcelp Encodage audio Qualcomm. qcp Andy Dejaco adejaco@qualcomm.com

audio/vnd.rhetorex. 32kadpcm Encodage audio Rhetorex™ ADPCM 32 kbit/s utilisé dans les solutions de messagerie vocale telles que CallPerformer™ et Unified Messenger™ de Lucent Technologies, entre autres. Jeff Bouis jbouis@lucent.com

audio/vnd.vmx.cvsd Encodage audio utilisé dans les produits de messagerie vocale, notamment Overture200™ de Lucent Technologies, Overture

Gammes de produits 300™ et VMX 300™. Identiques à celles présentées ci-dessus.

audio/x-aiff Format de fichier audio AIFF. aif, aiff, aifc

audio/x-pn-realaudio Format de métafichier RealAudio de Real Networks (anciennement Progressive Networks). ram, rm

audio/x-pn-realaudio-plugin Depuis Apache mime.types. rpm

Tableau D-4. Types MIME « audio » (suite)

Type MIME Description Extension Contact et référence

audio/x-realaudio Format audio RealAudio de Real Networks (anciennement Progressive Networks). ra

audio/x-wav Fichiers audio WAV. wav

chimique/*

Une grande partie des informations du tableau D-5 a été obtenue grâce à la « Chemical MIME Home Page » (http://www.ch.ic.ac.uk/chemime/).

Tableau D-5. Types MIME « chimiques »

Type MIME Description Extension Contact et référence

chimique/x-alchimie Format d'alchimie alc http://www.camsoft.com

chimique/x-cache-csf csf

chemical/x-cactvs-binary Format binaire CACTVS cbin http://cactvs.cit.nih.gov

chimique/x-cactvs-ascii Format ASCII CACTVS cascii http://cactvs.cit.nih.gov

chemical/x-cactvs-table Format de tableau CACTVS ctab http://cactvs.cit.nih.gov

chimique/x-cdx Fichier ChemDraw eXchange cdx http://www.camsoft.com

chimique/x-cerius MSI Cerius II format cer http://www.msi.com

chimique/x-chemdraw Fichier ChemDraw chm http://www.camsoft.com

Format d'échange cristallographique chimique/x-cif cif http://www.bernstein-plussons.com/software/rasmol/ http://ndbserver.rutgers.edu/NDB/ mmcif/examples/index.html

chimique/x-mmcif CIF macromoléculaire mcif Identique à ci-dessus

chimique/x-chem3d Chem3D format c3d http://www.camsoft.com

chimique/x-cmdf Format de données CrystalMaker cmdf http://www.crystalmaker.co.uk

Programme Compass chimique/x-compass du CPA Takahashi

chimique/x-crossfire fichier bsd Crossfire

chemical/x-cml Langage de balisage chimique cml http://www.xml-cml.org

chemical/x-csml Langage de balisage de style chimique csml, csm http://www.mdli.com

format de fichier CTX du groupe Gasteiger chemical/x-ctx ctx

chimique/x-cxf cxf

chimique/x-daylight-smiles Format Smiles smi http://www.daylight.com/dayhtml/ smiles/index.html

chimique/x-embl-dlnucléotide Format nucléotidique EMBL emb http://mercury.ebi.ac.uk

chemical/x-galactic-spc Format SPC pour les données spectrales et chromatographiques spc http://www.galactic.com/galactic/ Data/spcvue.htm

Tableau D-5. Types MIME « chimiques » (suite)

Type MIME Description Extension Contact et référence

chemical/x-gamess-input GAMESS Format d'entrée inp, gam http://www.msg.ameslab.gov/

GAMESS/Graphismes/

MacMolPlt.shtml

chimique/x-gaussien-entrée Format d'entrée gaussien gau http://www.mdli.com

Point de contrôle gaussien chimique/x-gaussien Format de point de contrôle gaussien fch, fchk http://products.camsoft.com

Cube gaussien chimique/x-gaussien (fonction d'onde) au format cub http://www.mdli.com

chimique/séquence x-gcg8 gcg

format ToGenBank chimique/x-genbank

chimique/x-isostar Bibliothèque IsoStar d'interactions intermoléculaires istr, ist http://www.ccdc.cam.ac.uk

chemical/x-jcamp-dx Format d'échange de données spectroscopiques JCAMP jdx, dx http://www.mdli.com

chimique/x-jjc-review-surface Fichiers de contour orbital Re_View3 rv3 http://www.brunel.ac.uk/depts/ chem/ch241s/re_view/rv3.htm

chemical/x-jjc-review-xyz Fichiers d'animation Re_View3 xyb http://www.brunel.ac.uk/depts/ chem/ch241s/re_view/rv3.htm

chemical/x-jjc-review-vib Re_View3 Fichiers de vibrations rv2, vib
http://www.brunel.ac.uk/depts/chem/ch241s/re_view/rv3.htm

Images cinétiques (structure des protéines) chimiques/x-kinemage kin
http://www.faseb.org/protein/ kinemages/MageSoftware.html

format de fichier MacMolecule mcm

chimique/x-macromodelinput MacroModel Mécanique moléculaire
mmd, mmod http://www.columbia.edu/cu/chemistry/

chimique/x-mdl-molfile MDL Molfile mol http://www.mdli.com

chemical/x-mdl-rdfile Fichier de données de réaction rd
http://www.mdli.com

chemical/x-mdl-rxnfile Format de réaction MDL rxn
http://www.mdli.com

Fichier de données de structure MDL sd http://www.mdli.com

Format graphique transportable MDL tgf http://www.mdli.com

chimique/x-mif mif

chemical/x-mol2 Représentation portable d'une molécule SYBYL mol2
http://www.tripos.com

Chemical/x-molconn-Z Format Molconn-Z b
http://www.eslc.vabiotech.com/molconn/molconnz.html

chemical/x-mopac-input Format d'entrée MOPAC mop
http://www.mdli.com

chimique/x-mopac-graph Format de graphique MOPAC gpt
http://products.camsoft.com

chimique/x-ncbi-asn1 asn (ancienne forme)

chimique/x-ncbi-asn1-binaire val

Banque de données sur les protéines chimiques/x-pdb pdb pdb
http://www.mdli.com

Tableau D-5. Types MIME « chimiques » (suite)

Type MIME Description Extension Contact et référence

chemical/x-swissprot Base de données de séquences protéiques
SWISS-PROT sw http://www.expasy.ch/spdbv/text/ download.htm

Accord de Versailles sur les matériaux et les normes vms
http://www.acolyte.co.uk/JISO/

chimique/x-vmd Dynamique moléculaire visuelle vmd
http://www.ks.uiuc.edu/Research/ vmd/

Format de fichier Xtelplot chimique/x-xtel xtel
http://www.recipnet.indiana.edu/ graphics/xtelplot/xtelplot.htm

Format d'animation de coordonnées chimiques/x-xyz xyz
http://www.mdli.com

image/*

Le tableau D-6 résume certains des types d’images couramment échangés par courrier électronique et HTTP.

Tableau D-6. Types MIME « Image »

Type MIME Description Extension Contact et référence

image/bmp Format d'image BMP Windows. bmp

image/cgm Computer Graphics Metafile (CGM) est une norme internationale pour le stockage portable et le transfert d'illustrations 2D. Alan Francis

AHFrancis@open.ac.uk

Voir ISO 8632:1992, IS 8632:1992 Amendement 1 (1994) et IS 8632:1992 Amendement 2 (1995)

image/g3fax Flux d'octets de télécopie G3. RFC 1494

image/gif Images GIF Compuserve. gif RFC 1341

image/ief ief RFC 1314

image/jpeg Images JPEG. jpeg,jpg,

jpe, jfif Projet de norme JPEG ISO 10918-1

CD

image/naplps Images NALPS (North American Presentation Layer Protocol Syntax). ANSI X3.110-1983 CSA T500-1983

image/png Images Portable Network Graphics (PNG). png Brouillon Internet draft-boutell-pngspec-04.txt, « Png (Portable

Spécifications des graphiques réseau

Version 1.0”

image/prs.btif Format utilisé par la Nations Bank pour la visualisation d'images BTIF de chèques et d'autres applications. btif, btf Arthur Rubin arthurr@crt.com

image/prs.pti Images codées PTI. pti Juern Laun juern.laun@gmx.de

http://server.hvzgymn.wn.schulebw.de/pti/

image/tiff Images TIFF. tiff, tif RFC 2302

Tableau D-6. Types MIME « Image » (suite)

Type MIME Description Extension Contact et référence

image/vnd.cns.inf2 Prend en charge les fonctionnalités applicatives disponibles sur la plateforme de services réseau TRILOGUE Infinity de Comverse Network Systems. Ann McLaughlin

Systèmes de réseau Comverse amclaughlin@comversens.com

image/vnd.dxf Fichiers CAO vectoriels DXF. dxf

image/vnd.fastbidsheet Une feuille FastBid contient une image raster ou vectorielle qui représente un dessin d'ingénierie ou d'architecture. fbs Scott Becker scottb@bxwa.com

image/vnd.fpx Images Kodak FlashPix. fpx Chris Wing format_change_request@kodak.

avec

http://www.kodak.com

image/vnd.fst Format d'image de FAST Search and Transfer. fst Arild Fuldseth

Arild.Fuldseth@fast.no

image/vnd.fujixerox.edmics-

Format d'image MMR Fuji Xerox EDMICS. MMR Masanori Onda

Masanori.Onda@fujixerox.co.jp

image/vnd.fujixerox.edmicsrlc Format d'image Fuji Xerox EDMICS RLC. rlc Identique à ci-dessus

Les fichiers MIX image/vnd.mix contiennent des données binaires en flux, utilisées pour représenter des images et des informations associées. Ils sont utilisés par les logiciels Microsoft PhotDraw et PictureIt. Saveen Reddy2 saveenr@microsoft.com

image/vnd.net-fpx Images Kodak FlashPix. Chris Wing format_change_request@kodak.

avec

http://www.kodak.com

image/vnd.wap.wbmp De Apache mime.types. wbmp

image/vnd.xiff Format d'image étendu utilisé par le logiciel Pagis. xif Steve Martin smartin@xis.xerox.com

image/x-cmu-raster De Apache mime.types. ras

Images génériques PBM image/x-portable-anymap. pnm Jeff Poskanzer

http://www.acme.com/software/ pbmplus/

image/x-portable-bitmap Images bitmap PBM. pbm Identique à ci-dessus

image/x-portable-graymap Images en niveaux de gris PBM. pgm Identique à ci-dessus

image/x-portable-pixmap Images couleur PBM. ppm Identique à ci-dessus

image/x-rgb Images RVB de Silicon Graphics. rgb

image/x-xbitmap Images bitmap du système X-Window. xbm

image/x-xpixmap Images couleur du système X-Window. xpm

image/x-xwindowdump Images de capture d'écran du système X-Window. xwd

message/*

Les messages sont des types composites utilisés pour communiquer des objets de données (par courrier électronique, HTTP ou autres protocoles de transport). Le tableau D-7 décrit les types de messages MIME courants.

Tableau D-7. Types MIME « Message »

Type MIME Description Extension Contact et référence

message/delivery-status message/dispositionnotification RFC 2298

message/corps externe RFC 1341

message/http RFC 2616

message/news Définit un moyen de transmettre des articles d'actualité par courrier électronique pour une lecture humaine. message/rfc822 n'est pas suffisant car les en-têtes d'actualité ont une sémantique au-delà de celle définie par la RFC 822. RFC 1036

message/partiel : permet la transmission fragmentée de corps jugés trop volumineux pour être envoyés directement par courrier électronique. RFC 1341

message/rfc822 Un message électronique complet. RFC 1341

message/s-http Messages HTTP sécurisés, une alternative au protocole HTTP sur SSL. RFC 2660

modèle/*

Le type MIME « model » est une extension enregistrée auprès de l'IETF. Il représente des modèles mathématiques de mondes physiques, destinés à la conception assistée par ordinateur (CAO) et aux graphiques 3D. Le tableau D-8 décrit certains formats de modèles.

Tableau D-8. Types MIME « modèles »

Type MIME Description Extension Contact et référence

modèle/iges La spécification initiale d'échange graphique (IGES) définit un format de données neutre qui permet l'échange numérique d'informations entre les systèmes de conception assistée par ordinateur (CAO). igs, iges RFC 2077

modèle/maille msh, maillage,

silo RFC 2077

model/vnd.dwf Fichiers CAO DWF. Jason Pratt

jason.pratt@autodesk.com

model/vnd.flatland.3dml Prend en charge les modèles 3DML pris en charge par les produits Flatland. 3dml,

3dm Michael Powers pow@flatland.com http://www.flatland.com

Tableau D-8. Types MIME « modèles » (suite)

Type MIME Description Extension Contact et référence

model/vnd.gdl model/vnd.gs-gdl Le langage de description géométrique (GDL) est un langage de définition d'objet paramétrique pour ArchiCAD par Graphisoft. gdl, gsm, win, dor, lmp,rsm, msm, ism Attila Babits ababits@graphisoft.hu http://www.graphisoft.com

model/vnd.gtw Modèles Gen-Trix. gtw Yutaka Ozaki yutaka_ozaki@gen.co.jp

model/vnd.mts Format de modèle MTS par Virtue. mts Boris Rabinovitch boris@virtue3d.com

model/vnd.parasolid.transmit.binary Fichier de modélisation binaire Parasolid. x_b http://www.ugsolutions.com/ products/parasolid/

model/vnd.parasolid.transmit.text Fichier de modélisation texte Parasolid. x_t http://www.ugsolutions.com/ products/parasolid/

model/vnd.vtu Format de modèle VTU par Virtue. vtu Boris Rabinovitch boris@virtue3d.com

Fichiers de format de langage de balisage de réalité virtuelle model/vrml. wrl, vrml RFC 2077

multipart/*

Les types MIME multiparties sont des objets composites contenant d'autres objets. Le sous-type décrit l'implémentation du packaging multipartie et le traitement des composants. Les types de médias multiparties sont résumés dans le tableau D-9.

Tableau D-9. Types MIME « multiparties »

Type MIME Description Extension Contact et référence

Multipart/alternative : le contenu est constitué d'une liste de représentations alternatives, chacune possédant son propre type de contenu. Le client peut sélectionner le composant le mieux pris en charge. RFC 1341

Les fichiers Apple Macintosh contiennent des « forfaits de ressources » et d'autres données de bureau décrivant le contenu réel du fichier. Ce contenu multipart contient les métadonnées Apple d'un côté et le contenu réel de l'autre. http://www.isi.edu/in-notes/iana/assignments/media-types/ multipart/appledouble

multipart/byteranges. Lorsqu'un message HTTP inclut le contenu de plusieurs plages, celles-ci sont transmises dans un objet « multipart/byteranges ». Ce type de média comprend deux parties ou plus, séparées par des limites MIME, chacune possédant ses propres champs Content-Type et ContentRange. RFC 2068

Tableau D-9. Types MIME « multiparties » (suite)

Type MIME Description Extension Contact et référence

multipart/digest Contient une collection de messages électroniques individuels, sous une forme facile à lire. RFC 1341

Multipart/encrypted utilise deux parties pour gérer le contenu chiffré cryptographiquement. La première partie contient les informations de contrôle nécessaires au déchiffrement des données du corps de la seconde partie et est étiquetée selon la valeur du paramètre de protocole. La seconde partie contient les données chiffrées de type application/octet-stream. RFC 1847

multipart/form-data Utilisé pour regrouper un ensemble de valeurs suite au remplissage d'un formulaire par un utilisateur. RFC 2388

multipart/header-set Sépare les données utilisateur des métadonnées descriptives arbitraires. http://www.isi.edu/in-notes/iana/assignments/media-types/ multipart/header-set

multipart/mixte Une collection d'objets. RFC 1341

multipart/parallel Syntaxiquement identique à multipart/mixed, mais toutes les parties sont destinées à être présentées simultanément, sur les systèmes capables de le faire. RFC 1341

multipart/related : destiné aux objets composés constitués de plusieurs parties interdépendantes. Les relations entre les parties les distinguent des autres types d'objets. Ces relations sont souvent représentées par des liens internes aux composants de l'objet qui référencent les autres composants. RFC 2387

multipart/report définit un type de conteneur général pour les rapports de courrier électronique de toute nature. RFC 1892

Multipart/signed utilise deux parties pour gérer le contenu signé cryptographiquement. La première partie contient le contenu, y compris ses en-têtes MIME. La seconde contient les informations nécessaires à la vérification de la signature numérique. RFC 1847

multipart/voice-message : fournit un mécanisme permettant de regrouper un message vocal dans un conteneur étiqueté conforme à VPIM v2. RFC 2421 et 2423

texte/*

Les types de supports texte contiennent des caractères et des informations de formatage potentielles. Le tableau D-10 résume les types MIME de texte.

Tableau D-10. Types MIME « Texte »

Type MIME Description Extension Contact et référence

text/calendar prend en charge la norme de calendrier et de planification iCalendar. RFC 2445

Feuilles de style en cascade texte/css. css RFC 2318

texte/répertoire : contient les données d'enregistrement d'une base de données d'annuaire, telle que LDAP. RFC 2425

Texte enrichi : texte formaté simple, prenant en charge les polices, les couleurs et l'espacement. Des balises de type SGML servent au début et à la fin du formatage. RFC 1896

texte/html Fichier HTML. html, htm RFC 2854

text/parityfec Correction d'erreur directe pour le texte diffusé dans un flux RTP. RFC 3009

texte/brut Texte brut ancien. asc, txt

text/prs.lines.tag Prend en charge les formulaires balisés, tels que ceux utilisés pour l'inscription par e-mail. tag, dsc John Lines

john@paladin.demon.co.uk http://www.paladin.demon.co.uk/tag-types/

text/rfc822-headers : permet de regrouper un ensemble d'en-têtes d'e-mail, par exemple lors de l'envoi de rapports d'échec. RFC 1892

texte/texte enrichi : ancienne forme de texte enrichi. Voir texte/enrichi. rtx RFC 1341

Le format RTF (Rich Text Format) est une méthode d'encodage de texte et d'images formatés pour le transfert entre applications. Ce format est largement pris en charge par les applications de traitement de texte sur MS-DOS, Windows, OS/2 et Macintosh.

text/sgml Fichiers de balisage SGML. sgml, sgmRFC1874

text/t140 Prend en charge le texte T.140 standardisé, tel qu'utilisé dans le multimédia RTP synchronisé. RFC 2793

TSV est une méthode populaire d'échange de données entre bases de données et spread-

Feuilles de calcul et traitements de texte. Il se compose d'un ensemble de lignes, dont les champs sont séparés par des tabulations. tsv http://www.isi.edu/in-notes/iana/assignments/media-types/text/tabseparated-values

text/uri-list Listes simples et commentées d'URL et d'URN utilisées par les résolveurs d'URN et toute autre application nécessitant la communication de listes d'URI en masse. uris, uri RFC 2483

Tableau D-10. Types MIME « Texte » (suite)

Type MIME Description Extension Contact et référence

Les fichiers ABC sont un format lisible par l'homme pour les partitions musicales. abc http://www.gre.ac.uk/ ~c.walshaw/abc/ http://home1.swipnet.se/ ~w-11382/abcbnf.htm

text/vnd.curl Fournit un ensemble de langages de définition de contenu interprétés par le plug-in d'exécution CURL. curl Tim Hodge thodge@curl.com

Les fichiers de script client CommonDM text/vnd.DMClientScript sont utilisés comme hyperliens vers des sites non http (tels que BYOND, IRC ou telnet) accessibles par l'application client Dream Seeker. dms Dan Bradley dan@dantom.com

http://www.byond.com/code/ref/

text/vnd.fly Fly est un préprocesseur de texte qui utilise une syntaxe simple pour créer une interface entre les bases de données et les pages Web. fly John-Mark Gurney jmg@flyidea.com http://www.flyidea.com

texte/vnd.fmi.flexstor À utiliser dans les projets SUVDAMA et UVRAPPF. flx http://www.ozone.fmi.fi/ SUVDAMA/

http://www.ozone.fmi.fi/UVRAPPF/

texte/vnd.in3d.3dml pour In3D Player. 3dml,

3dm Michael Powers powers@insideout.net

text/vnd.in3d.spot Pour In3D Player. spot, spo Identique à ci-dessus

text/vnd.IPTC.NewsML Format NewsML spécifié par l'International Press Telecommunications Council (IPTC). xmlDavid Allen

m_director_iptc@dial.pipex.com http://www.iptc.org

text/vnd.IPTC.NITF Format NITF spécifié par l'IPTC. xml Identique à ci-dessus http://www.nitf.org

text/vnd.latex-z Prend en charge les documents LaTeX contenant la notation Z. La notation Z (prononcée « zed ») est basée sur la théorie des ensembles de Zermelo-Fraenkel et la logique des prédicats du premier ordre. Elle est utile pour décrire les systèmes informatiques. http://www.comlab.ox.ac.uk/archive/z/

text/vnd.motorola.reflex : offre une méthode courante pour envoyer des messages texte simples depuis des appareils sans fil ReFLEX™. Mark Patton fmp014@email.mot.com

Une partie de la spécification FLEXsuite™ des protocoles d'activation est disponible

de Motorola dans le cadre de l'accord de licence

text/vnd.ms-mediapackage Ce type est destiné à être géré par les programmes d'application Microsoft MStore.exe et 7 storDB.exe. mpf Jan Nelson jann@microsoft.com

Tableau D-10. Types MIME « Texte » (suite)

Type MIME Description Extension Contact et référence

Les objets d'indication de service (SI) contiennent un message décrivant un événement et un URI décrivant où charger le service correspondant. si, xml WAP Forum Ltd http://www.wapforum.org

text/vnd.wap.sl Le type de contenu Service Loading (SL) fournit un moyen de transmettre un URI à un

Agent utilisateur dans un client mobile. Le client charge automatiquement le contenu indiqué par cet URI et l'exécute dans l'agent utilisateur concerné, sans intervention de l'utilisateur, le cas échéant. sl, xml : comme ci-dessus.

text/vnd.wap.wml Wireless Markup Language (WML) est un langage de balisage, basé sur XML, qui définit le contenu et l'interface utilisateur pour les appareils à bande étroite, y compris les téléphones portables et les téléavertisseurs. wml Identique à ci-dessus

text/vnd.wap.wmlscript WMLScript est une évolution de JavaScript pour les appareils sans fil. wmls Identique à ci-dessus

texte/x-setext De Apache mime.types. etx

Fichier au format text/xml Extensible Markup Language (utilisez application/xml si vous souhaitez que le navigateur enregistre dans un fichier lors du téléchargement). xml RFC 2376

vidéo/*

Le tableau D-11 répertorie quelques formats vidéo courants. Notez que certains formats vidéo sont classés comme types d'application. Tableau D-11. Types MIME « Vidéo »

Type MIME Description Extension Contact et référence

vidéo/MP4V-ES Charge utile vidéo MPEG-4, telle que transportée par RTP. RFC 3016

vidéo/mpeg Vidéo encodée selon la norme ISO 11172 CD MPEG. mpeg, mpg, mpe RFC 1341

video/parityfec Format vidéo de correction d'erreurs directes pour les données transmises via des flux RTP. RFC 3009

vidéo/pointeur : transport des informations de position du pointeur pour les présentations. RFC 2862

vidéo/quicktime Format vidéo Apple Quicktime. qt, mov http://www.apple.com

video/vnd.fvt Format vidéo de FAST Search & Transfer. fvt Arild Fuldseth

Arild.Fuldseth@fast.no

Tableau D-11. Types MIME « Vidéo » (suite)

Type MIME Description Extension Contact et référence

Formats propriétaires utilisés par les produits Motorola ISG. Tom McGinty, Motorola ISG, tmcginty@dma.isg.mot .

video/vnd.mpegurl Ce type de média se compose d'une série d'URL de fichiers vidéo MPEG. mxu Heiko Recktenwald uzs106@uni-bonn.de

« Pouvoir et responsabilité : conversations avec les contributeurs », Guy van

Belle et al., LMJ 9 (1999), 127-133, 129 (MIT Press)

video/vnd.nokia.interleavedmultimedia Utilisé dans le lecteur vidéo Nokia 9210 Communicator et les outils associés. nim Petteri Kangaslampi petteri.kangaslampi@nokia.com

vidéo/x-msvideo Films Microsoft AVI. avi http://www.microsoft.com

vidéo/x-sgi-movie Format de film de Silicon Graphics. film http://www.sgi.com

Types expérimentaux

L'ensemble des types principaux prend en charge la plupart des types de contenu. Le tableau D-12 répertorie un type expérimental, pour les logiciels de conférence, configuré sur certains serveurs Web.

Tableau D-12. Types MIME d'extension

Type MIME Description Extension Contact et référence

x-conference/x-cooltalk Outil de collaboration de Netscape ice

Annexe ECeci est le titre du livreANNEXE E

Codage en base 64

Le codage base-64 est utilisé par HTTP pour l'authentification de base et l'authentification Digest, ainsi que par plusieurs extensions HTTP. Cette annexe explique le codage base-64 et fournit des tables de conversion et des pointeurs vers des logiciels Perl pour vous aider à utiliser correctement le codage base-64 dans les logiciels HTTP.

Le codage Base-64 sécurise les données binaires

Le codage base-64 convertit une série d'octets arbitraires en une séquence plus longue de caractères de texte courants, qui constituent tous des valeurs de champ d'en-tête valides. Il permet de récupérer des données utilisateur ou binaires, de les compresser dans un format sécurisé et de les transmettre sous forme de valeurs de champ d'en-tête HTTP sans risquer qu'elles contiennent des deux-points, des sauts de ligne ou des valeurs binaires susceptibles de perturber les analyseurs HTTP.

Le codage Base-64 a été développé dans le cadre de la norme de messagerie électronique multimédia MIME, afin que MIME puisse transporter du texte enrichi et des données binaires arbitraires entre différentes passerelles de messagerie existantes.* Le codage Base-64 est similaire dans son esprit, mais plus efficace dans l'espace, aux normes uuencode et BinHex pour la texturation des données binaires. La section 6.8 de la RFC 2045 de MIME détaille l'algorithme base-64.

Huit bits à six bits

Le codage en base 64 prend une séquence d'octets de 8 bits, la décompose en segments de 6 bits et attribue à chaque segment de 6 bits l'un des 64 caractères composant l'alphabet en base 64. Les 64 caractères de sortie possibles sont courants et peuvent être placés en toute sécurité dans les champs d'en-tête HTTP. Ces 64 caractères incluent les lettres majuscules et minuscules, les chiffres, le signe +,

* Certaines passerelles de messagerie supprimaient silencieusement de nombreux caractères « non imprimables » dont les valeurs ASCII étaient comprises entre 0 et 31. D'autres programmes interprétaient certains octets comme des caractères de contrôle de flux ou des caractères de contrôle spéciaux, ou convertissaient les retours chariot en sauts de ligne, etc. Certains programmes rencontraient des erreurs fatales à la réception de caractères internationaux d'une valeur supérieure à 127, car le logiciel n'était pas « correct 8 bits ».

et /. Le caractère spécial = est également utilisé. L'alphabet en base 64 est présenté dans le tableau E-1.

Notez que comme le codage en base 64 utilise des caractères de 8 bits pour représenter 6 bits d'informations, les chaînes codées en base 64 sont environ 33 % plus grandes que les valeurs d'origine.

Tableau E-1. Alphabet en base 64

0 A 8 I 16 Q 24 Y 32 g 40 o 48 w 56 4

1 B 9 J 17 R 25 Z 33 h 41 p 49 x 57 5

2 C 10 K 18 S 26 a 34 i 42 q 50 y 58 6

3 D 11 L 19 T 27 b 35 j 43 r 51 z 59 7

4 E 12 L 20 U 28 c 36 k 44 s 52 0 60 8

5 F 13 N 21 V 29 d 37 l 45 t 53 1 61 9

6 G 14 O 22 W 30 e 38 m 46 u 54 2 62 +

7 H 15 P 23 X 31 f 39 n 47 v 55 3 63 /

La figure E-1 présente un exemple simple d'encodage en base 64. Ici, la valeur d'entrée de trois caractères « Aïe ! » est encodée en base 64, ce qui donne la valeur de quatre caractères « T3ch ». Le fonctionnement est le suivant :

1. La chaîne « Aïe ! » est divisée en 3 octets de 8 bits (0x4F, 0x77, 0x21).

2. Les 3 octets créent la valeur binaire 24 bits 010011110111011100100001.

3. Ces bits sont segmentés en séquences de 6 bits 010011, 110111, 01110,100001.

4. Chacune de ces valeurs de 6 bits représente un nombre compris entre 0 et 63, correspondant à l'un des 64 caractères de l'alphabet en base 64. La chaîne de 4 caractères codée en base 64 ainsi obtenue est « T3ch », qui peut ensuite être transmise sur le réseau sous forme de caractères 8 bits « sûrs », car seuls les caractères les plus portables sont utilisés (lettres, chiffres, etc.).

Personnages 8 bits Aïe !

Valeur 8 bits (hexadécimal) $4F$77$21

Valeur 8 bits (binaire) 010011110111011100100001

Valeur 6 bits (décimal) 19 55 2833

Caractère base 64 T 3 ch

Figure E-1. Exemple d'encodage en base 64

Huit bits à six bits

Rembourrage Base-64

Le codage en base 64 prend une séquence d'octets de 8 bits et segmente le flux binaire en segments de 6 bits. Il est peu probable que la séquence de bits se divise uniformément en segments de 6 bits. Si la séquence de bits ne se divise pas uniformément en segments de 6 bits, elle est complétée par des bits nuls à la fin afin que sa longueur soit un multiple de

24 (le plus petit multiple commun de 6 et 8 bits).

Lors de l'encodage de la chaîne de bits complétée, tout groupe de 6 bits entièrement complété (ne contenant aucun bit des données d'origine) est représenté par un 65e symbole spécial :

« = ». Si un groupe de 6 bits est partiellement complété, les bits de remplissage sont mis à zéro.

Le tableau E-2 présente des exemples de remplissage. La chaîne d'entrée initiale « a:a » fait 3 octets, soit 24 bits. 24 étant un multiple de 6 et 8, aucun remplissage n'est nécessaire. La chaîne résultante, codée en base 64, est « YTph ».

Tableau E-2. Exemples de remplissage en base 64

Données d'entrée Séquence binaire (remplissage noté « x ») Données codées

a:a 011000 010011 101001 100001 YTph

a:aa 011000 010011 101001 100001 011000 01xxxx xxxxxx xxxxxx YTphYQ==

a:aaa 011000 010011 101001 100001 011000 010110 0001xx xxxxxx YTphYWE=

a:aaaa 011000 010011 101001 100001 011000 010110 000101 100001 YTphYWFh

Cependant, lorsqu'un autre caractère est ajouté, la chaîne d'entrée atteint 32 bits. Le plus petit multiple de 6 et 8 étant de 48 bits, 16 bits de remplissage sont ajoutés. Les 4 premiers bits de remplissage sont mélangés à des bits de données. Le groupe de 6 bits résultant, 01xxxx, est traité comme 010000, 16 décimal ou encodé en base 64 Q. Les deux groupes de 6 bits restants sont tous des remplissages et sont représentés par « = ».

Implémentation de Perl

MIME::Base64 est un module Perl pour l'encodage et le décodage base64. Vous trouverez des informations sur ce module à l'adresse http://www.perldoc.com/perl5.6.1/lib/MIME/Base64.html.

Vous pouvez encoder et décoder des chaînes à l'aide des méthodes MIME::Base64 encode_base64 et decode_base64 :

```
utiliser MIME::Base64;

$encoded = encode_base64('Aladdin:Sésame, ouvre-toi');

$décodée = decode_base64($encodée);
```


Pour plus d'informations

Pour plus d'informations sur l'encodage en base 64, voir :

http://www.ietf.org/rfc/rfc2045.txt

La section 6.8 de la RFC 2045, « MIME Partie 1 : Format des corps de messages Internet », fournit une spécification officielle du codage en base 64.

http://www.perldoc.com/perl5.6.1/lib/MIME/Base64.html

Ce site Web contient la documentation du module Perl MIME::Base64 qui fournit l'encodage et le décodage des chaînes en base 64.

Pour plus d'informations

Annexe FCeci est le titre du livreANNEXE F

Authentification Digest

Cette annexe contient des données de support et du code source pour la mise en œuvre des fonctions d'authentification HTTP Digest.

Digest WWW-Authenticate Directives

Les directives WWW-Authenticate sont décrites dans le tableau F-1, paraphrasées à partir des descriptions de la RFC 2617. Comme toujours, reportez-vous aux spécifications officielles pour les détails les plus récents.

Tableau F-1. Directives d'en-tête Digest WWW-Authenticate (d'après la RFC 2617)

Description de la directive

royaume : une chaîne à afficher aux utilisateurs afin qu'ils sachent quel nom d'utilisateur et quel mot de passe utiliser. Cette chaîne doit contenir au moins le nom de l'hôte effectuant l'authentification et peut également indiquer le nombre d'utilisateurs pouvant y avoir accès. Par exemple, « registrated_users@gotham.news.com ».

nonce : une chaîne de données spécifiée par le serveur, qui doit être générée de manière unique à chaque réponse 401. Il est recommandé que cette chaîne soit en base 64 ou hexadécimale. Plus précisément, comme la chaîne est transmise entre guillemets dans les lignes d'en-tête, les guillemets doubles ne sont pas autorisés.

Le contenu du nonce dépend de l'implémentation. La qualité de l'implémentation dépend d'un choix judicieux. Un nonce peut, par exemple, être construit comme l'encodage en base 64 de : horodatage H(horodatage ":" ETag ":" clé privée)

Où l'horodatage est une heure générée par le serveur ou une autre valeur non répétitive, l'ETag est la valeur de l'en-tête HTTP ETag associé à l'entité demandée, et la clé privée est une donnée connue uniquement du serveur. Avec un nonce de cette forme, le serveur recalcule la partie hachage après réception de l'en-tête d'authentification du client et rejette la requête si elle ne correspond pas au nonce de cet en-tête ou si la valeur d'horodatage n'est pas suffisamment récente. De cette façon, le serveur peut limiter la durée de validité du nonce. L'inclusion de l'ETag empêche toute demande de relecture pour une version mise à jour de la ressource. (Remarque : l'inclusion de l'adresse IP du client dans le nonce semble permettre au serveur de limiter la réutilisation du nonce au client initial. Cependant, cela perturberait les fermes de proxys, où les requêtes d'un même utilisateur transitent souvent par différents proxys. De plus, l'usurpation d'adresse IP n'est pas si compliquée.)

Une implémentation peut choisir de ne pas accepter un nonce ou un condensé précédemment utilisé, pour se protéger contre les attaques par relecture, ou elle peut choisir d'utiliser des nonces ou des condensés à usage unique pour les requêtes POST ou PUT et des horodatages pour les requêtes GET.

Tableau F-1. Directives d'en-tête Digest WWW-Authenticate (d'après RFC 2617) (suite)

Description de la directive

domaine Une liste d'URI entre guillemets et séparée par des espaces (comme spécifié dans la RFC 2396, « Uniform Resource Identifiers : Generic

Syntaxe) qui définissent l'espace de protection. Si un URI est un chemin_abs, il est relatif à l'URL racine canonique du serveur auquel on accède. Un URI absolu de cette liste peut faire référence à un serveur différent de celui auquel on accède.

Le client peut utiliser cette liste pour déterminer l'ensemble des URI pour lesquels les mêmes informations d'authentification peuvent être envoyées : tout URI qui a un URI dans cette liste comme préfixe (après que les deux ont été rendus absolus) peut être supposé être dans le même espace de protection.

Si cette directive est omise ou si sa valeur est vide, le client doit supposer que l'espace de protection est constitué de tous les URI sur le serveur répondant.

Cette directive n'a pas de sens dans les en-têtes Proxy-Authenticate, pour lesquels l'espace de protection est toujours l'intégralité du proxy ; si elle est présente, elle doit être ignorée.

Opaque : une chaîne de données, spécifiée par le serveur, qui doit être renvoyée par le client sans modification dans l'en-tête d'autorisation des requêtes ultérieures dont les URI se trouvent dans le même espace de protection. Il est recommandé que cette chaîne soit en base 64 ou en hexadécimal.

stale : indicateur indiquant que la requête précédente du client a été rejetée car la valeur du nonce était obsolète. Si stale est VRAI (insensible à la casse), le client peut souhaiter réessayer la requête avec une nouvelle réponse chiffrée, sans demander à l'utilisateur un nouveau nom d'utilisateur et un nouveau mot de passe. Le serveur ne doit définir stale sur VRAI que s'il reçoit une requête pour laquelle le nonce est invalide mais dont le condensé est valide (indiquant que le client connaît le nom d'utilisateur et le mot de passe corrects). Si stale est FAUX, ou autre que VRAI, ou si la directive stale est absente, le nom d'utilisateur et/ou le mot de passe sont invalides et de nouvelles valeurs doivent être obtenues.

algorithme : chaîne indiquant une paire d'algorithmes utilisés pour générer le condensé et une somme de contrôle. En son absence, la valeur est supposée être « MD5 ». Si l'algorithme n'est pas compris, la requête doit être ignorée (et un autre, s'il y en a plusieurs).

Dans ce document, la chaîne obtenue en appliquant l'algorithme de digestion aux données « data » avec secret

« secret » sera noté « KD(secret, data) », et la chaîne obtenue en appliquant l'algorithme de somme de contrôle aux données « data » sera notée « H(data) ». La notation « unq(X) » correspond à la valeur de la chaîne « X » entre guillemets, sans les guillemets qui l'entourent.

Pour les algorithmes MD5 et MD5-sess :

H(données) = MD5(données)

HD(secret, données) = H(concat(secret, ":", données))

Autrement dit, le condensé est le MD5 du secret concaténé avec un deux-points concaténé aux données. L'algorithme MD5sess est conçu pour permettre l'utilisation de serveurs d'authentification tiers efficaces.

qop Cette directive est facultative mais est rendue ainsi uniquement pour des raisons de compatibilité descendante avec la RFC 2069 [6] ; elle doit être utilisée par toutes les implémentations conformes à cette version du schéma de résumé.

S'il est présent, il s'agit d'une chaîne entre guillemets d'un ou plusieurs jetons indiquant les valeurs de « qualité de protection » prises en

charge par le serveur. La valeur « auth » indique une authentification ; la valeur « auth-int » indique une authentification avec protection d'intégrité. Les options non reconnues doivent être ignorées.

<extension> Cette directive autorise les extensions futures. Toute directive non reconnue doit être ignorée.

Directives d'autorisation de résumé

Chacune des directives d'autorisation est décrite dans le tableau F-2, paraphrasé à partir des descriptions de la RFC 2617. Reportez-vous aux spécifications officielles pour les plus récentes.

détails.

Directives d'autorisation de résumé

Tableau F-2. Directives d'en-tête d'autorisation Digest (d'après la RFC 2617)

Description de la directive

nom d'utilisateur Le nom de l'utilisateur dans le domaine spécifié.

royaume Le royaume transmis au client dans l'en-tête WWW-Authenticate.

nonce Le même nonce transmis au client dans l'en-tête WWW-Authenticate.

uri L'URI de l'URI de demande de la ligne de demande ; dupliqué car les proxys sont autorisés à modifier la ligne de demande en transit, et nous pouvons avoir besoin de l'URI d'origine pour des calculs de vérification de résumé appropriés.

réponse Il s'agit du condensé réel, tout l'intérêt de l'authentification par condensé ! La réponse est une chaîne de 32 chiffres hexadécimaux, calculée par un algorithme de condensé négocié, qui prouve que l'utilisateur connaît le mot de passe.

algorithme : chaîne indiquant une paire d'algorithmes utilisés pour générer le condensé et une somme de contrôle. En son absence, la valeur est supposée être « MD5 ».

opaque Une chaîne de données, spécifiée par le serveur dans un en-tête WWW-Authenticate, qui doit être renvoyée par le client sans modification dans l'en-tête d'autorisation des requêtes ultérieures avec des URI dans le même espace de protection.

cnonce Cela doit être spécifié si une directive qop est envoyée et ne doit pas être spécifié si le serveur n'a pas envoyé de directive qop dans le champ d'en-tête WWW-Authenticate.

La valeur cnonce est une valeur de chaîne entre guillemets opaque fournie par le client et utilisée par le client et le serveur pour éviter les attaques en texte clair choisies, pour fournir une authentification mutuelle et pour fournir une certaine protection de l'intégrité des messages.

Voir les descriptions des calculs de résumé de réponse et de résumé de demande plus loin dans cette annexe.

qop : indique la « qualité de protection » appliquée au message par le client. Si cette valeur est présente, elle doit correspondre à l'une des alternatives acceptées par le serveur dans l'en-tête WWW-Authenticate. Ces valeurs affectent le calcul du résumé de la requête.

Il s'agit d'un jeton unique, et non d'une liste d'alternatives citées, comme dans WWW-Authenticate.

Cette directive est facultative, afin de préserver la compatibilité descendante avec une implémentation minimale de la RFC 2069, mais elle doit être utilisée si le serveur a indiqué que qop est pris en charge en fournissant une directive qop dans le champ d'en-tête WWW-Authenticate.

nc Ceci doit être spécifié si une directive qop est envoyée et ne doit pas être spécifié si le serveur n'a pas envoyé de directive qop dans le champ d'en-tête WWW-Authenticate.

La valeur correspond au nombre hexadécimal de requêtes (requête actuelle incluse) envoyées par le client avec la valeur de nonce de cette requête. Par exemple, lors de la première requête envoyée en réponse à une valeur de nonce donnée, le client envoie nc=“00000001”.

Le but de cette directive est de permettre au serveur de détecter les rediffusions de requêtes en conservant sa propre copie de ce décompte : si la même valeur nc est vue deux fois, la requête est une rediffusion.

<extension> Cette directive autorise les extensions futures. Toute directive non reconnue doit être ignorée.

Directives d'informations d'authentification Digest

Chacune des directives Authentication-Info est décrite dans le tableau F-3, paraphrasé à partir des descriptions de la RFC 2617. Reportez-vous aux spécifications officielles pour obtenir les détails les plus récents.

Tableau F-3. Directives d'en-tête Digest Authentication-Info (d'après la RFC 2617)

Description de la directive

nextnonce : la valeur de la directive nextnonce correspond au nonce que le serveur souhaite que le client utilise pour une future réponse d’authentification. Le serveur peut envoyer l’en-tête Authentication-Info avec un champ nextnonce afin d’implémenter des nonces

ponctuels ou variables. Si le champ nextnonce est présent, le client doit l'utiliser lors de la construction de l'en-tête Authorization de sa prochaine requête. Si le client ne le fait pas, une demande de réauthentification du serveur avec la valeur « stale=TRUE » peut être générée.

Les implémentations serveur doivent soigneusement prendre en compte les implications en termes de performances de l'utilisation de ce mécanisme ; les requêtes pipelinées ne seront pas possibles si chaque réponse inclut une directive nextnonce devant être utilisée lors de la prochaine requête reçue par le serveur. Il convient de prendre en compte le compromis entre performances et sécurité que représente l'autorisation d'utiliser une ancienne valeur de nonce pendant une durée limitée pour permettre le pipelining des requêtes. L'utilisation du compteur de nonce permet de conserver la plupart des avantages de sécurité d'un nouveau nonce serveur sans les effets néfastes sur le pipelining.

qop : indique les options de « qualité de protection » appliquées à la réponse par le serveur. La valeur « auth » indique une authentification ; la valeur « auth-int » indique une authentification avec protection d'intégrité. Le serveur doit utiliser la même valeur pour la directive qop dans la réponse que celle envoyée par le client dans la requête correspondante.

rspauth Le résumé de réponse facultatif de la directive « response auth » prend en charge l'authentification mutuelle : le serveur prouve qu'il connaît le secret de l'utilisateur et, avec qop=« auth-int », il assure également une protection limitée de l'intégrité de la réponse. La valeur de « response-digest » est calculée comme pour « request-digest » dans l'en-tête Authorization, sauf que si qop=« auth » ou qop n'est pas spécifié dans l'en-tête Authorization de la requête, A2 est :

A2 = ":" digest-uri-value et si qop="auth-int", A2 est :

A2 = ":" digest-uri-value ":" H(entité-corps)

où digest-uri-value correspond à la valeur de la directive uri de l'en-tête Authorization de la requête. Les valeurs cnonce et nc doivent être identiques à celles de la requête client à laquelle ce message répond. La directive rspauth doit être présente si qop="auth" ou qop="auth-int" est spécifié.

cnonce : la valeur cnonce doit être identique à celle de la requête client à laquelle ce message répond. La directive cnonce doit être présente si qop="auth" ou qop="auth-int" est spécifié.

La valeur nc doit être identique à celle de la requête client à laquelle ce message répond. La directive nc doit être présente si qop="auth" ou qop="auth-int" est spécifié.

<extension> Cette directive autorise les extensions futures. Toute directive non reconnue doit être ignorée.

Code de référence

Le code suivant implémente les calculs de H(A1), H(A2), request-digest et response-digest, selon la RFC 2617. Il utilise l'implémentation MD5 de la RFC 1321.

Fichier « digcalc.h »

#define HASHLEN 16 typedef char HASH[HASHLEN];

#define HASHHEXLEN 32

typedef char HASHHEX[HASHHEXLEN+1]; #define IN

Code de référence

#définir OUT

/* calculer H(A1) selon la spécification HTTP Digest */ void DigestCalcHA1(IN char * pszAlg,

DANS char * pszUserName,

DANS char * pszRealm,

DANS char * pszPassword,

DANS char * pszNonce,

DANS char * pszCNonce,

Clé de session OUT HASHHEX

);

/* calculer request-digest/response-digest selon la spécification HTTP Digest */ void DigestCalcResponse(

DANS HASHHEX HA1, /* H(A1) */

IN char * pszNonce, /* nonce du serveur */

IN char * pszNonceCount, /* 8 chiffres hexadécimaux */

DANS char * pszCNonce, /* client nonce */

DANS char * pszQop, /* qop-value: "", "auth", "auth-int" */

IN char * pszMethod, /* méthode de la requête */

DANS char * pszDigestUri, /* URL demandée */

DANS HASHHEX HEntity, /* H(corps d'entité) si qop="auth-int" */

OUT HASHHEX Réponse /* request-digest ou response-digest */

); Fichier « digcalc.c »

```
#include <global.h>

#include <md5.h>

#include <string.h>

#include "digcalc.h"

void CvtHex(

   DANS HASH Bin,

   HASHHEX Hex

   )

{

   i non signé court ; j non signé ; pour (i = 0 ; i < HASHLEN ; i++) { j =
(Bin[i] >> 4) & 0xf ; si (j <= 9)

         Hex[i*2] = (j + '0'); sinon

         Hex[i*2] = (j + 'a' - 10); j = Bin[i] & 0xf; si (j <= 9)

         Hex[i*2+1] = (j + '0'); sinon

         Hex[i*2+1] = (j + 'a' - 10);

   };

   Hex[HASHHEXLEN] = '\0';

};

/* calculer H(A1) selon les spécifications */ void DigestCalcHA1( IN char
* pszAlg,

   DANS char * pszUserName,

   DANS char * pszRealm,

   DANS char * pszPassword,

   DANS char * pszNonce,

   DANS char * pszCNonce,

   Clé de session OUT HASHHEX

   )

{

    MD5_CTX Md5Ctx;

    HACHAGE HA1;

    MD5Init(&Md5Ctx);

    MD5Update(&Md5Ctx, pszUserName, strlen(pszUserName));
```

```
    MD5Update(&Md5Ctx, ":", 1);
    MD5Update(&Md5Ctx, pszRealm, strlen(pszRealm));
    MD5Update(&Md5Ctx, ":", 1);
    MD5Update(&Md5Ctx, pszPassword, strlen(pszPassword));
    MD5Final(HA1, &Md5Ctx);
    si (stricmp(pszAlg, "md5-sess") == 0) {
        MD5Init(&Md5Ctx);
        MD5Update(&Md5Ctx, HA1, HASHLEN);
        MD5Update(&Md5Ctx, ":", 1);
        MD5Update(&Md5Ctx, pszNonce, strlen(pszNonce));
        MD5Update(&Md5Ctx, ":", 1);
        MD5Update(&Md5Ctx, pszCNonce, strlen(pszCNonce));
        MD5Final(HA1, &Md5Ctx);
    };
    CvtHex(HA1, SessionKey);
};
/* calculer request-digest/response-digest selon la spécification HTTP
Digest */ void DigestCalcResponse(
   DANS HASHHEX HA1, /* H(A1) */
   IN char * pszNonce, /* nonce du serveur */
   IN char * pszNonceCount, /* 8 chiffres hexadécimaux */
   DANS char * pszCNonce, /* client nonce */
   DANS char * pszQop, /* qop-value: "", "auth", "auth-int" */
   IN char * pszMethod, /* méthode de la requête */
   DANS char * pszDigestUri, /* URL demandée */
   DANS HASHHEX HEntity, /* H(corps d'entité) si qop="auth-int" */
   OUT HASHHEX Réponse /* request-digest ou response-digest */
   )
{
    MD5_CTX Md5Ctx;
    HACHAGE HA2;
```

```
    HASH RespHash;
     HASHHEX HA2Hex;
    // calculer H(A2)
    MD5Init(&Md5Ctx);
    MD5Update(&Md5Ctx, pszMethod, strlen(pszMethod));
    MD5Update(&Md5Ctx, ":", 1);
    MD5Update(&Md5Ctx, pszDigestUri, strlen(pszDigestUri)); si
(stricmp(pszQop, "auth-int") == 0) {
```

Code de référence

```
        MD5Update(&Md5Ctx, ":", 1);
        MD5Update(&Md5Ctx, HEntity, HASHHEXLEN);
    };
    MD5Final(HA2, &Md5Ctx);
     CvtHex(HA2, HA2Hex);
    // calculer la réponse
    MD5Init(&Md5Ctx);
    MD5Update(&Md5Ctx, HA1, HASHHEXLEN);
    MD5Update(&Md5Ctx, ":", 1);
    MD5Update(&Md5Ctx, pszNonce, strlen(pszNonce));
    MD5Update(&Md5Ctx, ":", 1); si (*pszQop) {
       MD5Update(&Md5Ctx, pszNonceCount, strlen(pszNonceCount));
       MD5Update(&Md5Ctx, ":", 1);
       MD5Update(&Md5Ctx, pszCNonce, strlen(pszCNonce));
       MD5Update(&Md5Ctx, ":", 1);
       MD5Update(&Md5Ctx, pszQop, strlen(pszQop));
       MD5Update(&Md5Ctx, ":", 1);
    };
    MD5Update(&Md5Ctx, HA2Hex, HASHHEXLEN);
    MD5Final(RespHash, &Md5Ctx);
    CvtHex(RespHash, Réponse);
};
```

Fichier « digtest.c »

```
#include <stdio.h>

#include "digcalc.h"

void main(int argc, char ** argv) { char * pszNonce =
"dcd98b7102dd2f0e8b11d0f600bfb0c093"; char * pszCNonce =
"0a4f113b"; char * pszUser = "Mufasa"; char * pszRealm = "
testrealm@host.com "; char * pszPass = "Cercle de la vie"; char *
pszAlg = "md5"; char szNonceCount[9] = "00000001"; char *
pszMethod = "GET"; char * pszQop = "auth"; char * pszURI =
"/dir/index.html";

    HASHHEX HA1;

    HASHHEX HA2 = "";

    Réponse HASHHEX ;

    DigestCalcHA1(pszAlg, pszUser, pszRealm, pszPass, pszNonce,
pszCNonce, HA1);

    DigestCalcResponse(HA1, pszNonce, szNonceCount, pszCNonce,
pszQop, pszMethod, pszURI, HA2, Réponse); printf("Réponse = %s\n",
Réponse);

};
```

Annexe GCeci est le titre du livre ANNEXE G

Balises de langue

Les balises de langue sont de courtes chaînes standardisées qui désignent les langues parlées, par exemple « fr » (français) et « en-GB » (anglais britannique). Chaque balise est composée d'une ou plusieurs parties, séparées par des tirets, appelées sous-balises. Les balises de langue sont décrites en détail dans la section « Balises de langue et HTTP » du chapitre 16.

Cette annexe résume les règles, les balises normalisées et les informations d'enregistrement des balises de langue. Elle contient les documents de référence suivants :

• Les règles pour la première sous-balise (principale) sont résumées dans « Règles de la première sous-balise ».

• Les règles de la deuxième sous-balise sont résumées dans « Règles de la deuxième sous-balise ».

• Les balises de langue enregistrées par l’IANA sont présentées dans le tableau G-1.

• Les codes de langue ISO 639 sont indiqués dans le tableau G-2.

• Les codes pays ISO 3166 sont indiqués dans le tableau G-3.

Règles de la première sous-étiquette

Si la première sous-étiquette est :

• Composé de deux caractères, il s'agit d'un code de langue issu des normes ISO 639* et 639-1

• Composé de trois caractères, il s'agit d'un code de langue répertorié dans la norme ISO 639-2†

• La lettre « i », la balise de langue est explicitement enregistrée auprès de l'IANA

• La lettre « x », la balise de langue, est une sous-balise d'extension privée et non standard. Les noms ISO 639 et 639-2 sont résumés dans le tableau G-2.

* Voir la norme ISO 639, « Codes pour la représentation des noms de langues ».

† Voir ISO 639-2, « Codes pour la représentation des noms de langues — Partie 2 : Code alpha-3 ».

Règles de la deuxième sous-étiquette

Si la deuxième sous-étiquette est :

• Deux caractères de long, il s'agit d'un pays/région défini par la norme ISO 3166*

• De trois à huit caractères, il peut être enregistré auprès de l'IANA

• Un seul caractère de long, c'est illégal

Les codes pays ISO 3166 sont résumés dans le tableau G-3.

Balises de langue enregistrées par l'IANA

Tableau G-1. Balises de langue

Description de la balise de langue IANA

i-bnn Bunun

i-default Contexte de langue par défaut

i-hak Hakka

i-klingon Klingon

i-lux luxembourgeois

i-mingo Mingo

i-navajo Navajo

i-pwn Paiwan

i-tao Tao

i-tay Tayal

i-tsu Tsou

norvégien no-bok « langue du livre »

no-nyn norvégien « nouveau norvégien »

zh-gan Kan ou Gan

zh-guoyu mandarin ou chinois standard

zh-hakka Hakka

zh-min Min, Fuzhou, Hokkien, Amoy ou taïwanais

zh-wuu Shanghaien ou Wu

zh-xiang Xiang ou Hunanais

zh-yue cantonais

* Les codes pays AA, QM–QZ, XA–XZ et ZZ sont réservés par la norme ISO 3166 comme codes attribués par l'utilisateur. Ils ne doivent pas être utilisés pour former des balises de langue.

Codes de langue ISO 639

Tableau G-2. Codes de langue ISO 639 et 639-2

Langue ISO 639 ISO 639-2

Abkhaze ab abk

Un as chinois

Acoli ach

Adangme ada

Afar aa aar

Afrihili afh

Afrikaans de afr

Afro-asiatique (Autre) afa

Akan alias

Akkadien akk

Albanais carré alb/sqi

bière aléoute

Langues algonquiennes alg

Altaïque (Autre) tut

amharique am amh

Langages Apache apa

Arabe ar ara

arc araméen

Arapaho arp

Araucanie

Arawak arw

Arménien hy arm/hye

Art artificiel (autre)

Assamais comme asm

Langues athapascanes ath

Carte austronésienne (autre)

Avaric ava

Avenue Avestan

Awadhi awa

Aymara ay aym

az aze azerbaïdjanais

Aztèque non

interdiction balinaise

Chauve-souris baltique (autre)

Langue ISO 639 ISO 639-2

Baloutche

Bambara bam

Langues bamilékées bai

Banda bad

Bantous (Autre) bnt

Basa bas

Bachkir ba bak

Basque eu baq/eus

Beja bej

Bemba bem

bengali bn ben

Berbère (Autre) ber

Bhojpuri bho

Bihari bh bih

Bikol bik

Bini bin

Bichlamar bi bis

Soutien-gorge Braj

Breton être bre

punaise buginaise

bulgare bg bul

bouriate bua

Birman my bur/mya

Biélorusse être bel

Caddo cad

Voiture des Caraïbes

chat catalan ca

Caucasien (Autre) cau

Cebuano ceb

Celtique (Autre) cel

Indien d'Amérique centrale (autre) cai

Chagatai chg

Chamorro cha

Tchétchène che

Cherokee chr

Cheyenne chy

Langue ISO 639 ISO 639-2

Chibcha chb

chinois zh chi/zho

Jargon chinook chn

Choctaw cho

slave d'église chu

Tchouvache chv

policier copte

Cor de Cornouailles

Co cos corses

Cree cre

Ruisseau mus

Créoles et pidgins (autres) crp

Créoles et pidgins, anglais (autre) cpe

Créoles et pidgins, basés sur le français (Autres) cpf

Créoles et pidgins, à base portugaise (Autres) cpp

Couchitique (autre) cus

hr croate

Tchèque cs ces/cze

Dakota dak

Danois da dan

Delaware del

Dinka din

Divehi div

Dogri doi

Dravidien (Autre) dra

Douala dua

néerlandais nl dut/nla

Néerlandais, moyen-âge (vers 1050-1350)

Dyula dyu

Dzongkha dz dzo

Efik efi

Égypte (ancienne)

Ekajuk eka

Elamite elx

Anglais en eng

Anglais, Moyen Âge (vers 1100-1500) enm

Langue ISO 639 ISO 639-2

Anglais, ancien (ca. 450-1100) ang

Esquimau (Autre) esk

Espéranto eo epo

Estonien et est

Brebis brebis

Ewondo ewo

Fan de crocs

Fanti gras

Féroïen fo fao

Fidjien fj fij

Finnois fi fin

Finno-ougrien (autre) fiu

Fon fon

Français fr fra/fre

Français, Moyen Âge (vers 1400-1600)

Français, ancien (842- environ 1400) de

Frison fy fry

Peul

Ga gaa

Gaélique (écossais) gae/gdh

Gallegan gl glg

Ganda lug

Gayo gay

Mon Dieu, mon Dieu

Géorgien ka geo/kat

Allemand de deu/ger

Allemand, Moyen-Haut (vers 1050-1500) gmh

Allemand, ancien haut (vers 750-1050) goh

Gemme germanique (autre)

gil gilbertais

Gondi gon

Gothique a

Grebo grb

Grec ancien (jusqu'en 1453) grc

Grec, moderne (1453-) el ell/gre

kl kal groenlandais

Langue ISO 639 ISO 639-2

Guarani gn grn

Gujarati gu guj

Haïda

Haoussa ha hau

Hawaïenne

hébreu he heb

Herero elle

Hiligaynon hil

Himachali lui

Salut Hindi

Hiri Motu hmo

Hongrois hu hun

Hupa hop

Iban iba

L'islandais est « glace/île »

Igbo ibo

Ijo, ijo

Iloko ilo

Indic (Autre) inc.

indo-européen (autre) ine

Indonésien id ind

Interlingua (IALA) ia ina

Interlingue ie ine

Inuktitut iu iku

Inupiak ik ipk

Iranien (Autre) ira

Irlandais ga gai/iri

Irlandais, ancien (jusqu'à 900) sga

Irlandais, Moyen (900 - 1200) mga

Langues iroquoiennes iro

italien it ita

Japonais ja jpn

Javanais jv/jw jav/jaw

Judéo-arabe jrb

judéo-persan jpr

Kabyle

Langue ISO 639 ISO 639-2

Kachin kac

Kamba kam

Kannada kn kan

Kanuri kau

Kara-Kalpak kaa

Karen Kar

Cachemire ks kas

Kawi kaw

Kazakh kk kaz

Khasi kha

Khmer km khm

Khoisan (Autre) khi

Kho khotanais

Kikuyu kik

Kinyarwanda rw kin

Kirghize ky kir

Komi kom

Kongo kon

Konkani kok

ko kor coréen

Kpelle kpe

Kru kro

Kuanyama kua

Kumyk kum

Kurde ku kur

Kurukh kru

Kusaie kus

Kutenai kut

Garçon ladino

Lahnda lah

Lamba lam

Langue d'Oc (après 1500) oc oci

Lao-lo-lao

latin lat

lav letton

Luxembourgeois ltz

Langue ISO 639 ISO 639-2

Lezghien

Lingala ln lin

lituanien lt lit

Lozi loz

Luba-Katanga lub

Luiseno lui

Lunda Lun

Luo (Kenya et Tanzanie) luo

Macédonien mk mac/mak

Madurais fou

Magahi mag

Maithili mai

Makasar mak

Malgache mg mlg

malais ms may/msa

Malayalam mal

Maltais ml mlt

Homme mandingue

Manipuri mni

Langues manobo mno

Manx max

Maori mi mao/IRM

Marathi M. Mar

Mari chm

Marshall Mah

Marwari mwr

Masaï mas

Langues mayas myn

Hommes Mende

Micro Micmac

Minangkabau min

Divers (Autres) mis

Mohawk moh

Mo mol moldave

Mon-Kmer (Autre) mkh

Mongo mdr

Langue ISO 639 ISO 639-2

Mongol mn mon

Mossi mos

Plusieurs langues mul

Langues Munda mun

Nauru na nau

Navajo nav

Ndebele, Nord nde

Ndebele, Sud nbl

Ndongo ndo

Ne nep népalais

Newari nouveau

Niger-Kordofanien (Autre) nic

Nilo-saharienne (autre) ssa

Niu niué

Norrois, ancien non

Indien d'Amérique du Nord (autre) nai

Norvégien non ni

Norvégien (Nynorsk) nno

Langues nubiennes nub

Nyamwezi nym

Nyanja nya

Nyankole nyn

Nyoro nyo

Nzima nzi

Ojibwa oji

Oriya ou ori

Oromo om orm

Osage osa

ossète

Langues otomiennes oto

Pahlavi pal

Pau palaosien

Pali pli

Pampanga pam

Pangasinan pag

Panjabi pa pan

Langue ISO 639 ISO 639-2

Papiamento pap

Papouan-Australien (Autre) paa

Persan fa fas/per

Persan, peuple ancien (environ 600 - 400 av. J.-C.)

Phénicien phn

Polonais pl pol

Ponape pon

Portugais pt por

Langues prakrites pra

Provençal, ancien (jusqu'à 1500) pro

Ps pus pachto

Quechua qu que

Rhéto-roman rm roh

Rajasthan

Rarotongan rar

Romance (Autre) roa

ro ron/rhum roumain

Roms roms

Rundi rn run

russe ru rus

Langues salishanes sal

araméen samaritain sam

Langues sami smi

Samoan sm smo

Sandawe triste

Sango sg sag

Sanskrit sa san

Sarde srd

Écossais sco

Selkup sel

Sémitique (Autre) sem

Serbe sr

serbo-croate sh scr

Sérère srr

Shan shn

Shona sn sna

Langue ISO 639 ISO 639-2

Sidamo sid

Siksika bla

Sindhi sd snd

Singhalais si sin

Sino-tibétain (autre) assis

Langues sioux sio

Slave (autre) sla

Siswant ss ssw

Slovaque sk slk/slo

Slovène sl slv

Sogdien sog

Somalien so som

fils songhaï

Langues sorabes wen

Sotho, NSO du Nord

Sotho, rue du Sud

Indien d'Amérique du Sud (autre) sai

Espagnol es esl/spa

Sukuma suk

Sumérien sux

Soudanais su sun

Susu sus

Swahili sw swa

Swazi ssw

Suédois sv sve/swe

syriaque syr

Tagalog tl tgl

tah tahitien

Tadjik tg tgk

Tamashek tmh

Tamoul ta tam

Tatar tt tat

Telugu te tel

Tereno ter

Thaï th tha

Bo bod/tib tibétain

Langue ISO 639 ISO 639-2

Tigre tig

Tigrinya ti tir

Temps de réponse

Tivi tiv

Tlingit tli

Tonga (Nyasa) à tog

Tonga (îles Tonga) ton

Truk tru

Tsimshian tsi

Tsonga ts tso

Tswana tn tsn

Tumbuka tum

Turc tr tur

Turc, ottoman (1500–1928) ota

Turkmène tk tuk

Tyv tuvinien

Twi tw twi

Uga ougaritique

Ouïghour oug uig

Ukrainien Royaume-Uni UKR

Umbundu umb

Indéterminé et

Ourdou ur urd

Ouzbek uz uzb

Vai vai

Venda ven

Vi vie vietnamien

Volapuk vo vol

Vote votique

Langues wakashanes wak

Walamo wal

Guerre de Waray

Washo était

Gallois cy cym/wel

Wolof wo wol

Xhosa xh xho

Langue ISO 639 ISO 639-2

Yakut sah

Yao Yao

Yap yap

Yiddish yi yid

Yoruba yo yo

Zap zapotèque

Zenaga zen

Zhuang za zha

Zoulou zu zul

Zuni Zun

Codes pays ISO 3166

Tableau G-3. Codes pays ISO 3166

Code du pays

Afghanistan AF

Albanie AL

Algérie DZ

Samoa américaines AS

Andorre AD

Angola AO

IA d'Anguilla

Antarctique AQ

Antigua-et-Barbuda AG

Argentine AR

Arménie AM

Aruba AW

Australie AU

Autriche AT

Azerbaïdjan AZ

Bahamas BS

Bahreïn BH

Bangladesh BD

Barbade BB

Biélorussie PAR

Belgique BE

Code du pays

Belize BZ

Bénin BJ

Bermudes BM

Bhoutan BT

Bolivie BO

Licence en Bosnie-Herzégovine

Botswana BW

Île Bouvet BV

Brésil BR

Territoire britannique de l'océan Indien IO

Brunéi Darussalam BN

Bulgarie BG

Burkina Faso BF

Burundi BI

Cambodge KH

CM du Cameroun

Canada CA

CV du Cap-Vert

Îles Caïmans KY

République centrafricaine CF

Tchad TD

Chili CL

Chine CN

Île Christmas CX

Îles Cocos (Keeling) CC

Colombie CO

Comores KM

Congo CG

Congo (République démocratique du) CD

Îles Cook CK

Costa Rica CR

Côte d'Ivoire CI

Croatie HR

Cuba CU

Chypre CY

République tchèque CZ

Code du pays

Danemark DK

DJ djiboutien

Dominique DM

République dominicaine DO

Timor oriental TP

Équateur CE

Égypte EG

El Salvador SV

Guinée équatoriale GQ

Érythrée ER

Estonie EE

Éthiopie ET

Îles Falkland (Malvinas) FK

Îles Féroé FO

Fidji FJ

Finlande FI

France FR

Guyane française GF

Polynésie française PF

Terres australes françaises TF

Gabon GA

Directeur général de la Gambie

Géorgie GE

Allemagne DE

Ghana GH

Gibraltar GI

Grèce GR

Groenland GL

Grenade GD

GP de Guadeloupe

Guam GU

Guatemala GT

Guinée GN

Guinée-Bissau GW

Guyane GY

Haïti HT

Code du pays

Îles Heard et McDonald HM

Saint-Siège (État de la Cité du Vatican) VA

Honduras HN

Hong Kong HK

Hongrie HU

Islande IS

Inde IN

Carte d'identité indonésienne

Iran (République islamique d') RI

QI en Irak

Irlande IE

Israël IL

Italie IT

Jamaïque JM

Japon JP

Jordan JO

Kazakhstan KZ

Kenya KE

Kiribati KI

Corée (République populaire démocratique de) KP

Corée (République de) KR

Koweït KW

Kirghizistan KG

République démocratique populaire lao LA

Lettonie LV

Liban LB

Lesotho LS

Libéria LR

Jamahiriya arabe libyenne LY

Liechtenstein LI

Lituanie LT

Luxembourg LU

Macao MO

Macédoine (ex-République yougoslave de) MK

Madagascar MG

Malawi MW

Code du pays

Malaisie MY

Maldives MV

Mali ML

Malte MT

Îles Marshall MH

Martinique MQ

Mauritanie MR

Maurice MU

Mayotte (YT)

Mexique MX

Micronésie (États fédérés de) FM

Moldavie (République de) MD

Monaco MC

Mongolie MN

Montserrat MS

Maroc MA

Mozambique MZ

Myanmar MM

Namibie NA

Nauru NR

Parc national du Népal

Pays-Bas NL

Antilles néerlandaises AN

Nouvelle-Calédonie NC

Nouvelle-Zélande NZ

Nicaragua NI

Niger NE

Nigéria NG

Niue NU

Île Norfolk (Terre-Neuve)

député des Îles Mariannes du Nord

Norvège NON

Oman OM

Pakistan PK

Palau PW

Territoire palestinien (occupé) PS

Panama PA

Code du pays

Papouasie-Nouvelle-Guinée PG

Paraguay PY

Pérou PE

Philippines PH

Pitcairn PN

Pologne PL

Portugal PT

Porto Rico RP

Contrôle qualité du Qatar

Réunion RE

Roumanie RO

Fédération de Russie RU

Rwanda RW

Sainte-Hélène SH

Saint-Kitts-et-Nevis KN

Sainte-Lucie LC

Premier ministre de Saint-Pierre-et-Miquelon

Saint-Vincent-et-les Grenadines VC

Samoa WS

Saint-Marin SM

Sao Tomé-et-Principe ST

Arabie saoudite SA

Sénégal SN

Seychelles SC

Sierra Leone SL

Singapour SG

Slovaquie SK

Slovénie SI

Îles Salomon SB

Somalie SO

Afrique du Sud ZA

GS de la Géorgie du Sud et des îles Sandwich du Sud

Espagne ES

Sri Lanka LK

Soudan SD

Suriname SR

Svalbard et Jan Mayen SJ

Code du pays

Swaziland SZ

Suède SE

Suisse CH

République arabe syrienne SY

Taïwan, province de Chine TW

Tadjikistan TJ

Tanzanie (République-Unie de) TZ

Thaïlande TH

Togo TG

Tokelau TK

Tonga TO

Trinité-et-Tobago TT

Tunisie TN

Turquie TR

Turkménistan TM

Îles Turques-et-Caïques TC

Télévision de Tuvalu

Ouganda UG

Ukraine UA

Émirats arabes unis AE

Royaume-Uni GB

États-Unis

Îles mineures éloignées des États-Unis (UM)

Uruguay UY

Ouzbékistan UZ

Vanuatu VU

Venezuela VE

Vietnam VN

Îles Vierges (britanniques) VG

ÎLES Vierges (États-Unis) VI

Wallis et Futuna WF

Sahara occidental EH

Yémen YE

Yougoslavie YU

Zambie ZM

Organisations administratives linguistiques

La norme ISO 639 définit une agence de maintenance pour les ajouts et les modifications à la liste des langues de la norme ISO 639. Cette agence est :

Centre international d'information terminologique (Infoterm)

Boîte postale 130 A-1021 Vienne

Autriche

Téléphone : +43 1 26 75 35 poste 312

Fax : +43 1 216 32 72

La norme ISO 639-2 définit une agence de maintenance pour les ajouts et les modifications à la liste des langues de la norme ISO 639-2. Cette agence est :

Bibliothèque du Congrès

Bureau du développement du réseau et des normes MARC

Washington, DC 20540

USA

Téléphone : +1 202 707 6237

Télécopieur : +1 202 707 0115

URL : http://www.loc.gov/standards/iso639/

L'organisme de maintenance de la norme ISO 3166 (codes pays) est :

Secrétariat de l'agence de maintenance ISO 3166 c/o DIN Deutsches Institut fuer Normung

Burggrafenstrasse 6

Case postale 1107

D-10787 Berlin

Allemagne

Téléphone : +49 30 26 01 320

Fax : +49 30 26 01 231

URL : http://www.din.de/gremien/nas/nabd/iso3166ma/

Organisations administratives linguistiques

Annexe HCeci est le titre du livreANNEXE H

Registre des jeux de caractères MIME

Cette annexe décrit le registre des jeux de caractères MIME géré par l'IANA (Internet Assigned Numbers Authority). Un tableau formaté des jeux de caractères du registre est fourni dans le tableau H-1.

Registre des jeux de caractères MIME

Les balises de jeu de caractères MIME sont enregistrées auprès de l'IANA (http://www.iana.org/numbers.htm). Le registre des jeux de caractères est une base de données textuelle à plat. Chaque enregistrement contient un nom de jeu de caractères, des citations de référence, un numéro MIB unique, une description de la source et une liste d'alias. Un nom ou un alias peut être signalé comme « nom MIME préféré ». Voici l'enregistrement pour l'US-ASCII :

Nom : ANSI_X3.4-1968 [RFC1345, KXS2]^

MIBenum: 3

Source : registre ECMA

Alias : iso-ir-6

Alias : ANSI_X3.4-1986

Alias : ISO_646.irv:1991

Alias : ASCII

Alias : ISO646-US

Alias : US-ASCII (nom MIME préféré)

Alias: nous

Alias : IBM367

Alias : cp367

Alias : csASCII

La procédure d'enregistrement d'un jeu de caractères auprès de l'IANA est documentée dans la RFC 2978

(http://www.ietf.org/rfc/rfc2978.txt).

Noms MIME préférés

Sur les 235 jeux de caractères enregistrés au moment de la rédaction de cet article, seuls 20 incluent des « noms MIME préférés », des jeux de caractères courants utilisés par les applications de messagerie et web. Il s'agit des suivants :

Big5 EUC-JP EUC-KR

GB2312 ISO-2022-JP ISO-2022-JP-2

ISO-2022-KR ISO-8859-1 ISO-8859-2

ISO-8859-3 ISO-8859-4 ISO-8859-5

ISO-8859-6 ISO-8859-7 ISO-8859-8

ISO-8859-9 ISO-8859-10 KOI8-R

Maj-JIS US-ASCII

Jeux de caractères enregistrés

Le tableau H-1 répertorie le contenu du registre des jeux de caractères en mars 2001. Reportez-vous directement à http://www.iana.org pour plus d'informations sur le contenu de ce tableau.

Tableau H-1. Balises de jeu de caractères MIME IANA

Balise de jeu de caractères Alias Description Références

US-ASCII ANSI_X3.4-1968, iso-ir-6,

ANSI_X3.4-1986,

ISO_646.irv:1991, ASCII, ISO646-US, us, IBM367, cp367, csASCII
Registre ECMA RFC1345, KXS2

ISO-10646-UTF-1 csISO10646UTF1 Format de transfert universel (1) — il s'agit du codage multi-octets qui sous-ensemble ASCII-7 ; il ne présente pas de problèmes d'ordre d'octets

ISO_646.basic:1983 ref, csISO646basic1983 Registre ECMA RFC1345, KXS2

INVARIANT csINVARIANT RFC1345, KXS2

ISO_646.irv : 1983 iso-ir-2, irv, csISO2IntlRefVersion Registre ECMA RFC1345, KXS2

BS_4730 iso-ir-4, ISO646-GB, gb, uk, csISO4UnitedKingdom Registre ECMA RFC1345, KXS2

NATS-SEFI iso-ir-8-1, registre csNATSSEFI ECMA RFC1345, KXS2

NATS-SEFI-ADD iso-ir-8-2, csNATSSEFIADD Registre ECMA RFC1345, KXS2

NATS-DANO iso-ir-9-1, csNATSDANO Registre ECMA RFC1345, KXS2

NATS-DANO-ADD iso-ir-9-2, csNATSDANOADD Registre ECMA RFC1345, KXS2

Balise de jeu de caractères Alias Description Références

SEN_850200_B iso-ir-10, FI, ISO646-FI, ISO646-SE, se, csISO10Registre ECMA suédois RFC1345, KXS2

SEN_850200_C iso-ir-11, ISO646-SE2, se2, csISO11SwedishForNames Registre ECMA RFC1345, KXS2

KS_C_5601-1987 iso-ir-149, KS_C_5601-1989, KSC_5601, coréen, csKSC56011987 registre ECMA RFC1345, KXS2

ISO-2022-KR csISO2022KR RFC 1557 (voir aussi KS_C_5601-1987) RFC1557, Choi

EUC-KR csEUCKR RFC 1557 (voir aussi KS_C_5861-1992) RFC1557, Choi

ISO-2022-JP csISO2022JP RFC 1468 (voir aussi RFC 2237) RFC1468,

Murai

ISO-2022-JP-2 csISO2022JP2 RFC 1554 RFC1554, Ohta

ISO-2022-CN RFC 1922 RFC1922

ISO-2022-CN-EXT RFC 1922 RFC1922

JIS_C6220-1969-jp JIS_C6220-1969, iso-ir-13, katakana, x0201-7, csISO13JISC6220jp Registre ECMA RFC1345, KXS2

JIS_C6220-1969-ro iso-ir-14, jp, ISO646-JP, csISO14JISC6220ro Registre ECMA RFC1345, KXS2

IT iso-ir-15, ISO646-IT, csISO15Registre ECMA italien RFC1345, KXS2

PT iso-ir-16, ISO646-PT, csISO16Registre ECMA portugais RFC1345, KXS2

ES iso-ir-17, ISO646-ES, csISO17Registre ECMA espagnol RFC1345, KXS2

greek7-old iso-ir-18, csISO18Greek7Old registre ECMA RFC1345, KXS2

latin-grec iso-ir-19, csISO19latingrec registre ECMA RFC1345, KXS2

DIN_66003 iso-ir-21, de, ISO646-DE, csISO21Registre ECMA allemand RFC1345, KXS2

NF_Z_62-010_(1973) iso-ir-25, ISO646-FR1, csISO25Registre ECMA français RFC1345, KXS2

Latin-grec-1 iso-ir-27, csISO27LatinGreek1 Registre ECMA RFC1345, KXS2

ISO_5427 iso-ir-37, csISO5427Cyrillique Registre ECMA RFC1345, KXS2

JIS_C6226-1978 iso-ir-42,

csISO42JISC62261978 Registre ECMA RFC1345, KXS2

BS_viewdata iso-ir-47, csISO47BSViewdata Registre ECMA RFC1345, KXS2

INIS iso-ir-49, csISO49Registre INIS ECMA RFC1345, KXS2

INIS-8 iso-ir-50, csISO50INIS8 Registre ECMA RFC1345, KXS2

INIS-cyrillique iso-ir-51, csISO51INISCyrillique registre ECMA RFC1345, KXS2

Balise de jeu de caractères Alias Description Références

ISO_5427:1981 iso-ir-54, ISO5427Cyrillique1981 Registre ECMA RFC1345, KXS2

ISO_5428 : 1980 iso-ir-55, csISO5428Registre ECMA grec RFC1345, KXS2

GB_1988-80 iso-ir-57, cn, ISO646-CN, csISO57GB1988 Registre ECMA RFC1345, K5,

KXS2

GB_2312-80 iso-ir-58, chinois, csISO58GB231280 Registre ECMA RFC1345, KXS2

NS_4551-1 iso-ir-60, ISO646-NO, non, csISO60DanishNorwegian, csISO60Norwegian1 Registre ECMA RFC1345, KXS2

NS_4551-2 ISO646-NO2, iso-ir-61, no2, csISO61Norwegian2 Registre ECMA RFC1345, KXS2

NF_Z_62-010 iso-ir-69, ISO646-FR, fr, csISO69Registre ECMA français RFC1345, KXS2

vidéotex-suppl iso-ir-70,

csISO70VideotexSupp1 Registre ECMA RFC1345, KXS2

PT2 iso-ir-84, ISO646-PT2, csISO84Portuguese2 Registre ECMA RFC1345, KXS2

Registre ES2 iso-ir-85, ISO646-ES2, csISO85Spanish2 ECMA RFC1345, KXS2

MSZ_7795.3 iso-ir-86, ISO646-HU, hu, csISO86Registre ECMA hongrois RFC1345, KXS2

JIS_C6226-1983 iso-ir-87, x0208, JIS_X0208-1983, csISO87JISX0208 Registre ECMA RFC1345, KXS2

greek7 iso-ir-88, csISO88Greek7 Registre ECMA RFC1345, KXS2

ASMO_449 ISO_9036, arabe7, iso-ir-89, csISO89ASMO449 Registre ECMA RFC1345, KXS2

iso-ir-90 csISO90 Registre ECMA RFC1345, KXS2

JIS_C6229-1984-a iso-ir-91, jp-ocr-a, csISO91JISC62291984a Registre ECMA RFC1345, KXS2

JIS_C6229-1984-b iso-ir-92, ISO646-JP-OCR-B, jp-ocr-b,

csISO92JISC62991984b Registre ECMA RFC1345, KXS2

Registre ECMA JIS_C6229-1984-b-add iso-ir-93, jp-ocr-b-add, csISO93JIS62291984badd RFC1345, KXS2

JIS_C6229-1984-main iso-ir-94, jp-ocr-main, csISO94JIS62291984main registre ECMA RFC1345, KXS2

JIS_C6229-1984-hand-add iso-ir-95, jp-ocr-hand-add, csISO95JIS62291984handadd registre ECMA RFC1345, KXS2

JIS_C6229-1984-kana iso-ir-96,

csISO96JISC62291984kana Registre ECMA RFC1345, KXS2

Balise de jeu de caractères Alias Description Références

ISO_2033-1983 iso-ir-98, e13b, csISO2033 Registre ECMA RFC1345, KXS2

ANSI_X3.110-1983 iso-ir-99, CSA_T500-1983, NAPLPS, csISO99NAPLPS registre ECMA RFC1345, KXS2

ISO-8859-1 ISO_8859-1:1987, iso-ir-100,

ISO_8859-1, latin1, l1,

IBM819, CP819, csISOLatin1 Registre ECMA RFC1345, KXS2

ISO-8859-2 ISO_8859-2:1987, iso-ir-101, ISO_8859-2, latin2, l2, csISOLatin2 Registre ECMA RFC1345, KXS2

T.61-7bit iso-ir-102, csISO102T617bit registre ECMA RFC1345, KXS2

T.61-8 bits T.61, iso-ir-103, csISO103T618 bits Registre ECMA RFC1345, KXS2

ISO-8859-3 ISO_8859-3:1988, iso-ir-109, ISO_8859-3, latin3, l3, csISOLatin3 Registre ECMA RFC1345, KXS2

ISO-8859-4 ISO_8859-4:1988, iso-ir-110, ISO_8859-4, latin4, l4, csISOLatin4 Registre ECMA RFC1345, KXS2

ECMA-cyrillique iso-ir-111, csISO111ECMACyrillique registre ECMA RFC1345, KXS2

CSA_Z243.4-1985-1 iso-ir-121, ISO646-CA, csa7-1, ca, csISO121Canadien1 Registre ECMA RFC1345, KXS2

CSA_Z243.4-1985-2 iso-ir-122, ISO646-CA2, csa7-2, csISO122Canadian2 Registre ECMA RFC1345, KXS2

CSA_Z243.4-1985-gr iso-ir-123,

csISO123CSAZ24341985gr Registre ECMA RFC1345, KXS2

ISO-8859-6 ISO_8859-6:1987, iso-ir-127,

ISO_8859-6, ECMA-114, ASMO-708, arabe, csISOLatinArabic Registre ECMA RFC1345, KXS2

ISO_8859-6-E csISO88596E RFC 1556 RFC1556, IANA

ISO_8859-6-I csISO88596I RFC 1556 RFC1556, IANA

ISO-8859-7 ISO_8859-7:1987, iso-ir-126,

ISO_8859-7, ELOT_928, ECMA-118, grec, greek8, csISOLatinGreek Registre ECMA RFC1947,

RFC1345, KXS2

T.101-G2 iso-ir-128, csISO128T101G2 registre ECMA RFC1345, KXS2

ISO-8859-8 ISO_8859-8:1988, iso-ir-138, ISO_8859-8, hébreu, csISOLatinHebrew Registre ECMA RFC1345, KXS2

ISO_8859-8-E csISO88598E RFC 1556 RFC1556,

Nussbacher

Balise de jeu de caractères Alias Description Références

ISO_8859-8-I csISO88598I RFC 1556 RFC1556,

Nussbacher

CSN_369103 iso-ir-139,

csISO139CSN369103 Registre ECMA RFC1345, KXS2

JUS_I.B1.002 iso-ir-141, ISO646-YU, js, yu, csISO141JUSIB1002 Registre ECMA RFC1345, KXS2

ISO_6937-2-add iso-ir-142, registre csISOTextComm ECMA et ISO 6937-2:1983 RFC1345, KXS2

IEC_P27-1 iso-ir-143, csISO143IECP271 registre ECMA RFC1345, KXS2

ISO-8859-5 ISO_8859-5:1988, iso-ir-144, ISO_8859-5, cyrillique, csISOLatinCyrillic Registre ECMA RFC1345, KXS2

JUS_I.B1.003-serb iso-ir-146, serbe, csISO146Registre ECMA serbe RFC1345, KXS2

JUS_I.B1.003-mac macédonien, iso-ir-147, csISO147Registre ECMA macédonien RFC1345, KXS2

ISO-8859-9 ISO_8859-9:1989, iso-ir-148, ISO_8859-9, latin5, l5, csISOLatin5 Registre ECMA RFC1345, KXS2

grec-ccitt iso-ir-150, csISO150, csISO150GreekCCITT registre ECMA RFC1345, KXS2

NC_NC00-10:81 Cuba, iso-ir-151, ISO646-CU, csISO151Cuba Registre ECMA RFC1345, KXS2

ISO_6937-2-25 iso-ir-152, csISO6937Ajouter le registre ECMA RFC1345, KXS2

GOST_19768-74 ST_SEV_358-88, iso-ir-153, csISO153GOST1976874 Registre ECMA RFC1345, KXS2

ISO_8859-supp iso-ir-154, latin1-2-5, csISO8859Supp registre ECMA RFC1345, KXS2

ISO_10367-box iso-ir-155, csISO10367Box registre ECMA RFC1345, KXS2

ISO-8859-10 iso-ir-157, l6,

ISO_8859-10:1992, csISOLatin6, registre ECMA latin6 RFC1345, KXS2

tour latin, iso-ir-158, csISO158Lap Registre ECMA RFC1345, KXS2

JIS_X0212-1990 x0212, iso-ir-159, csISO159JISX02121990 Registre ECMA RFC1345, KXS2

DS_2089 DS2089, ISO646-DK, dk, csISO646Danish Norme danoise, DS 2089, février

1974 RFC1345, KXS2

us-dk csUSDK RFC1345, KXS2

dk-us csDKUS RFC1345, KXS2

JIS_X0201 X0201, csHalfWidthKatakana JIS X 0201-1976 — 1 octet seulement ; cela équivaut à JIS/Roman (similaire à ASCII) plus katakana demi-largeur 8 bits RFC1345, KXS2

Balise de jeu de caractères Alias Description Références

KSC5636 ISO646-KR, csKSC5636 RFC1345, KXS2

ISO-10646-UCS-2 csUnicode Le plan multilingue de base à 2 octets,

également appelé Unicode : il doit spécifier l'ordre des octets du réseau ; la norme ne le spécifie pas (il s'agit d'un espace entier de 16 bits).

ISO-10646-UCS-4 csUCS4 L'espace de code complet (même commentaire

à propos de l'ordre des octets ; ceux-ci sont de 31 bits

Nombres)

Manuel d'utilisation du DEC-MCS dec, csDECMCS VAX/VMS, commande

Numéro : AI-Y517A-TE, avril 1986 RFC1345, KXS2

hp-roman8 roman8, r8, csHPRoman8 Manuel d'utilisation de l'imprimante LaserJet IIP, référence HP 33471-90901, Hewlett-

Packard, juin 1989 HP-PCL5,

RFC1345, KXS2

Macintosh Mac, csMacintosh La norme Unicode v1.0, ISBN

0201567881, octobre 1991 RFC1345, KXS2

IBM037 cp037, ebcdic-cp-us, ebcdic-cp-ca, ebcdic-cp-wt, ebcdic-cp-nl, csIBM037 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM038 EBCDIC-INT, cp038, csIBM038 Référence du jeu de caractères IBM 3174, GA27-

3831-02, mars 1990 RFC1345, KXS2

IBM273 CP273, csIBM273 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM274 EBCDIC-BE, CP274, csIBM274 IBM 3174 Jeu de caractères de référence, GA27-

3831-02, mars 1990 RFC1345, KXS2

IBM275 EBCDIC-BR, cp275, csIBM275 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM277 EBCDIC-CP-DK,EBCDIC-CP-NO, csIBM277 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM278 CP278, ebcdic-cp-fi, ebcdic-cp-se, csIBM278 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM280 CP280, ebcdic-cp-it, csIBM280 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM281 EBCDIC-JP-E, cp281, csIBM281 IBM 3174 Jeu de caractères de référence, GA27-

3831-02, mars 1990 RFC1345, KXS2

IBM284 CP284, ebcdic-cp-es, csIBM284 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM285 CP285, ebcdic-cp-gb, csIBM285 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM290 cp290, EBCDIC-JP-kana, csIBM290 IBM 3174 Jeu de caractères de référence, GA27-

3831-02, mars 1990 RFC1345, KXS2

IBM297 cp297, ebcdic-cp-fr, csIBM297 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

Balise de jeu de caractères Alias Description Références

IBM420 cp420, ebcdic-cp-ar1, csIBM420 IBM NLS RM Vol2 SE09-8002-01, mars

1990, IBM NLS RM p 11-11 RFC1345, KXS2

IBM423 cp423, ebcdic-cp-gr, csIBM423 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM424 cp424, ebcdic-cp-he, csIBM424 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM437 cp437, 437, csPC8CodePage437 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM500 CP500, ebcdic-cp-be, ebcdic-cp-ch, csIBM500 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

Guide de comparaison IBM775 cp775, csPC775Baltic HP PCL 5 (P/N 5021-

0329) pp B-13, 1996 HP-PCL5

IBM850 cp850, 850, csPC850, IBM NLS RM multilingue Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM851 cp851, 851, csIBM851 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM852 cp852, 852, csPCp852 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM855 cp855, 855, csIBM855 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM857 cp857, 857, csIBM857 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM860 cp860, 860, csIBM860 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM861 cp861, 861, cp-is, csIBM861 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM862 cp862, 862, csPC862LatinHébreu IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

Dispositions de clavier et pages de codes IBM863 cp863, 863, csIBM863,

PN 07G4586, juin 1991 RFC1345, KXS2

Dispositions de clavier et pages de codes IBM864 cp864, csIBM864 IBM,

PN 07G4586, juin 1991 RFC1345, KXS2

IBM865 cp865, 865, csIBM865 IBM DOS 3.3 Ref (abrégé), 94X9575,

Février 1987 RFC1345, KXS2

IBM866 cp866, 866, csIBM866 IBM NLDG Vol2 SE09-8002-03, août

Étang de 1994

IBM868 CP868, cp-ar, csIBM868 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

Dispositions de clavier et pages de codes IBM869 cp869, 869, cp-gr, csIBM869,

PN 07G4586, juin 1991 RFC1345, KXS2

IBM870 CP870, ebcdic-cp-roece, ebcdic-cp-yu, csIBM870 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

Balise de jeu de caractères Alias Description Références

IBM871 CP871, ebcdic-cp-is, csIBM871 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM880 cp880, EBCDIC-Cyrillique, csIBM880 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM891 cp891, csIBM891 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM903 cp903, csIBM903 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM904 cp904, 904, csIBBM904 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM905 CP905, ebcdic-cp-tr, csIBM905 IBM 3174 Jeu de caractères Référence, GA27-

3831-02, mars 1990 RFC1345, KXS2

IBM918 CP918, ebcdic-cp-ar2, csIBM918 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

IBM1026 CP1026, csIBM1026 IBM NLS RM Vol2 SE09-8002-01, mars

1990 RFC1345, KXS2

EBCDIC-AT-DE csIBMEBCDICATDE IBM 3270 Char Set Ref Ch 10, GA27-

2837-9, avril 1987 RFC1345, KXS2

EBCDIC-AT-DE-A csEBCDICATDEA IBM 3270 Char Set Ref Ch 10, GA27-

2837-9, avril 1987 RFC1345, KXS2

EBCDIC-CA-FR csEBCDICCAFR IBM 3270 Char Set Ref Ch 10, GA27-

2837-9, avril 1987 RFC1345, KXS2

EBCDIC-DK-NO csEBCDICDKNO IBM 3270 Jeu de caractères Réf. Ch 10, GA27-

2837-9, avril 1987 RFC1345, KXS2

EBCDIC-DK-NO-A csEBCDICDKNOA IBM 3270 Jeu de caractères Réf. Ch 10, GA27-

2837-9, avril 1987 RFC1345, KXS2

EBCDIC-FI-SE csEBCDICFISE IBM 3270 Jeu de caractères Réf. Ch 10, GA27-

2837-9, avril 1987 RFC1345, KXS2

EBCDIC-FI-SE-A csEBCDICFISEA IBM 3270 Char Set Ref Ch 10, GA27-

2837-9, avril 1987 RFC1345, KXS2

EBCDIC-FR csEBCDICFR IBM 3270 Jeu de caractères Réf. Ch 10, GA27-

2837-9, avril 1987 RFC1345, KXS2

EBCDIC-IT csEBCDICIT IBM 3270 Jeu de caractères Réf. Ch. 10, GA27-

2837-9, avril 1987 RFC1345, KXS2

EBCDIC-PT IBM 3270 Jeu de caractères Réf. Ch. 10, GA27-

2837-9, avril 1987 RFC1345, KXS2

EBCDIC-ES csEBCDICES IBM 3270 Jeu de caractères Réf. Ch. 10, GA27-

2837-9, avril 1987 RFC1345, KXS2

EBCDIC-ES-A csEBCDICESA IBM 3270 Jeu de caractères Réf. Ch. 10, GA27-

2837-9, avril 1987 RFC1345, KXS2

Balise de jeu de caractères Alias Description Références

EBCDIC-ES-S csEBCDICESS IBM 3270 Jeu de caractères Réf. Ch. 10, GA27-

2837-9, avril 1987 RFC1345, KXS2

EBCDIC-UK csEBCDICUK IBM 3270 Jeu de caractères Réf. Ch 10, GA27-

2837-9, avril 1987 RFC1345, KXS2

EBCDIC-US csEBCDICUS IBM 3270 Jeu de caractères Réf. Ch. 10, GA27-

2837-9, avril 1987 RFC1345, KXS2

INCONNU-8BITS csUnknown8BiT RFC1428

MNEMONIC csMnemonic RFC 1345, également connu sous le nom de

« mnémonique+ascii+38 » RFC1345, KXS2

MNEM csMnem RFC 1345, également connu sous le nom de

« mnémonique+ascii+8200 » RFC1345, KXS2

VISCII csVISCII RFC 1456 RFC1456

VIQR csVIQR RFC 1456 RFC1456

KOI8-R csKOI8R RFC 1489, basé sur GOST-19768-74,

ISO-6937/8, INIS-cyrillique, ISO-5427 RFC1489

KOI8-U RFC 2319 RFC2319

IBM00858 CCSID00858, CP00858, PC-

IBM multilingue à 850 euros (voir .../assignments/characterset-info/IBM00858) [Mahdi]

IBM00924 CCSID00924, CP00924, ebcdic-

Latin9--euro IBM (voir .../assignments/characterset-info/IBM00924) [Mahdi]

IBM01140 CCSID01140, CP01140, ebcdicus-37+euro IBM (voir .../assignments/characterset-info/IBM01140) [Mahdi]

IBM01141 CCSID01141, CP01141, ebcdicde-273+euro IBM (voir .../assignments/characterset-info/IBM01141) [Mahdi]

IBM01142 CCSID01142, CP01142, ebcdicdk-277+euro, ebcdic-no277+euro IBM (voir .../assignments/characterset-info/IBM01142) [Mahdi]

IBM01143 CCSID01143, CP01143, ebcdicfi-278+euro, ebcdic-se278+euro IBM (voir .../assignments/characterset-info/IBM01143) [Mahdi]

IBM01144 CCSID01144, CP01144, ebcdicit-280+euro IBM (voir .../assignments/characterset-info/IBM01144) [Mahdi]

IBM01145 CCSID01145, CP01145, ebcdices-284+euro IBM (voir .../assignments/characterset-info/IBM01145) [Mahdi]

IBM01146 CCSID01146, CP01146, ebcdicgb-285+euro IBM (voir .../assignments/characterset-info/IBM01146) [Mahdi]

IBM01147 CCSID01147, CP01147, ebcdicfr-297+euro IBM (voir .../assignments/characterset-info/IBM01147) [Mahdi]

IBM01148 CCSID01148, CP01148, ebcdicinternational-500+euro IBM (voir .../assignments/characterset-info/IBM01148) [Mahdi]

Balise de jeu de caractères Alias Description Références

IBM01149 CCSID01149, CP01149, ebcdicis-871+euro IBM (voir .../assignments/characterset-info/IBM01149) [Mahdi]

Big5-HKSCS Aucun Voir (.../assignments/character-setinfo/Big5-HKSCS) [Beurk]

UNICODE-1-1 csUnicode11 RFC 1641 RFC1641

SCSU Aucun SCSU (voir .../assignments/characterset-info/SCSU) [Scherer]

UTF-7 Aucun RFC 2152 RFC2152

UTF-16BE Aucun RFC 2781 RFC2781

UTF-16LE Aucun RFC 2781 RFC2781

UTF-16 Aucun RFC 2781 RFC2781

UNICODE-1-1-UTF-7 csUnicode11UTF7 RFC 1642 RFC1642

UTF-8 RFC 2279 RFC2279

iso-8859-13 ISO (voir ...assignments/character-setinfo/iso-8859-13)[Tumasonis]

iso-8859-14 iso-ir-199,

ISO_8859-14:1998, ISO_8859-14, latin8, iso-celtique, l8 ISO (voir ...assignments/character-setinfo/iso-8859-14) [Simonsen]

ISO-8859-15 ISO_8859-15 ISO

JIS_Encoding csJISEncoding JIS X 0202-1991 ; utilise les séquences d'échappement ISO 2022 pour décaler les ensembles de codes, comme documenté dans JIS X 0202-1991

Shift_JIS MS_Kanji, csShiftJIS Ce jeu de caractères est une extension de csHalfWidthKatakana : il ajoute des caractères graphiques conformes à la norme JIS X 0208. Les CCS sont JIS X0201:1997 et JIS X0208:1997. La définition complète est présentée à l'annexe 1 de la norme JISX0208:1997. Ce jeu de caractères peut être utilisé pour le type de média de niveau supérieur « texte ».

EUC-JP Extended_UNIX_Code_

Format_emballé_pour_le_japonais,

csEUCPkdFmtJapanese normalisé par OSF, UNIX

International et UNIX Systems Laboratories Pacific. Utilise les règles ISO 2022 pour sélectionner le jeu de codes. Jeu de codes 0 : USASCII (un seul jeu d'octets de 7 bits) ; Jeu de codes 1 : JIS X0208-1990 (un double jeu d'octets de 8 bits) limité à A0–FF dans les deux octets ; Jeu de codes 2 : katakana demi-chasse (un seul jeu d'octets de 7 bits) nécessitant SS2 comme préfixe de caractère ; Jeu de codes 3 : JIS X0212-1990 (un double jeu d'octets de 7 bits) limité à A0–FF dans les deux octets nécessitant SS3 comme préfixe de caractère.

Balise de jeu de caractères Alias Description Références

Code_UNIX_étendu_

Largeur_fixe_pour_

Japonais csEUCFixWidJapanese, utilisé au Japon. Chaque caractère est 2.

octets. ensemble de codes 0 : US-ASCII (un seul ensemble d'octets de 7 bits), 1er octet = 00, 2e octet =

20–7E ; jeu de codes 1 : JIS X0208-1990 (un jeu d'octets double de 7 bits) limité à A0–FF dans les deux octets ; jeu de codes 2 : katakana demi-largeur (un jeu d'octets simple de 7 bits), 1er

octet = 00, 2e octet = A0–FF ; ensemble de codes

3 : JIS X0212-1990 (un ensemble d'octets double de 7 bits) limité à A0–FF dans le premier octet et 21–7E dans le deuxième octet.

ISO-10646-UCS-Basic csUnicodeASCII Sous-ensemble ASCII d'Unicode. Latin de base = collection 1. Voir ISO 10646, Annexe A.

ISO-10646-Unicode-Latin1 csUnicodeLatin1, ISO-10646 Sous-ensemble ISO Latin-1 d'Unicode. Latin de base et Latin-1. Supplément = collections 1 et 2. Voir ISO 10646, Annexe A et RFC 1815.

ISO-10646-J-1 ISO 10646 japonais. Voir RFC 1815.

ISO-Unicode-IBM-1261 csUnicodeIBM1261 IBM Latin-2, -3, -5, étendu

Coffret de présentation, GCSGID : 1261

ISO-Unicode-IBM-1268 csUnidoceIBM1268 Ensemble de présentation étendu IBM Latin-4,

GCSGID : 1268

ISO-Unicode-IBM-1276 csUnicodeIBM1276 IBM Cyrillic Greek Extended

Coffret de présentation, GCSGID : 1276

ISO-Unicode-IBM-1264 csUnicodeIBM1264 Ensemble de présentations IBM arabe, GCSGID :

1264

ISO-Unicode-IBM-1265 csUnicodeIBM1265 Ensemble de présentation IBM en hébreu, GCSGID :

1265

ISO-8859-1-Windows-3.0-

Latin-1 csWindows30Latin1 ISO 8859-1 étendu Latin-1 pour

Windows 3.0. ID du jeu de symboles PCL : 9U. HP-PCL5

ISO-8859-1-Windows-3.1-

Latin-1 csWindows31Latin1 étendu ISO 8859-1 Latin-1 pour

Windows 3.1. ID du jeu de symboles PCL : 19U. HP-PCL5

ISO-8859-2-Windows-

Latin-2 csWindows31Latin2 Extended ISO 8859-2. Latin-2 pour

Windows 3.1. ID du jeu de symboles PCL : 9E. HP-PCL5

ISO-8859-9-Windows-

Latin-5 csWindows31Latin5 étendu ISO 8859-9. Latin-5 pour

Windows 3.1. ID du jeu de symboles PCL : 5T. HP-PCL5

Référence du langage PostScript Adobe Standard Encoding

Manuel. ID du jeu de symboles PCL : 10J. Adobe

Ventura-US csVenturaUS Ventura US-ASCII plus caractères généralement utilisés dans l'édition, tels que pilcrow, copyright, enregistré,

marque, section, poignard et double poignard dans la plage A0 (hex) à

FF (hex). ID du jeu de symboles PCL : 14J. HP-PCL5

Balise de jeu de caractères Alias Description Références

Ventura-International csVenturaInternational Ventura International. Caractères codés ASCII plus similaires à Roman8. ID du jeu de symboles PCL : 13J. HP-PCL5

PC8-Danois-Norvégien csPC8DanishNorwegian PC Jeu de symboles PCL 8 bits pour le danois et le norvégien.

11U. HP-PCL5

PC8-Turc csPC8Turkish PC Latin Turc. ID du jeu de symboles PCL : 9T. HP-PCL5

Ensemble de présentations IBM-Symbols csIBMSymbols, CPGID : 259 IBM-CIDT

Ensemble de présentation IBM-Thai csIBMThai, CPGID : 838 IBM-CIDT

Guide de comparaison HP-Legal csHPLegal PCL 5, HewlettPackard, référence HP 5961-0510, octobre 1992. ID du jeu de symboles PCL : 1U. HP-PCL5

Guide de comparaison des polices HP-Pi csHPPiFont PCL 5, HewlettPackard, numéro de pièce HP 5961-0510,

Octobre 1992. ID du jeu de symboles PCL : 15U. HP-PCL5

Guide de comparaison HP-Math8 csHPMath8 PCL 5, HewlettPackard, numéro de pièce HP 5961-0510,

Octobre 1992. ID du jeu de symboles PCL : 8 M. HP-PCL5

Référence du langage PostScript Adobe-Symbol-Encoding csHPPSMath

Manuel. ID du jeu de symboles PCL : 5M. Adobe

Guide de comparaison HP-DeskTop csHPDesktop PCL 5, Hewlett-Packard, référence HP 5961-0510, octobre 1992. ID du jeu de symboles PCL : 7J. HP-PCL5

Guide de comparaison Ventura-Math csVenturaMath PCL 5, HewlettPackard, numéro de pièce HP 5961-0510,

Octobre 1992. ID du jeu de symboles PCL : 6 M. HP-PCL5

Guide de comparaison Microsoft Publishing PCL 5, Hewlett-Packard, référence HP 5961-0510, octobre 1992. ID du jeu de symboles PCL : 6J. HP-PCL5

Windows-31J csWindows31J Windows japonais. Une extension supplémentaire de Shift_JIS pour inclure les caractères spéciaux NEC (ligne 13), la sélection NEC des extensions IBM (lignes 89 à 92) et les extensions IBM (lignes 115 à

119). Les normes CCS sont JIS X0201:1997, JIS X0208:1997 et les extensions suivantes. Ce jeu de caractères peut être utilisé pour le type de média de niveau supérieur « texte », mais son utilisation est limitée ou spécialisée (voir RFC 2278). ID du jeu de symboles PCL : 19 Ko.

GB2312 csGB2312 Chinois pour la République populaire de Chine

(PRC) ensemble mixte 1 octet, 2 octets : 20–7E

= ASCII 1 octet ; A1–FE = Kanji PRC 2 octets. Voir GB 2312-80. ID du jeu de symboles PCL : 18C.

Balise de jeu de caractères Alias Description Références

Ensemble multi-octets Big5 csBig5 chinois pour Taïwan. ID de l'ensemble de symboles PCL : 18T.

windows-1250 Microsoft (voir .../character-set-info/ windows-1250) [Lazhintseva]

windows-1251 Microsoft (voir .../character-set-info/ windows-1251) [Lazhintseva]

windows-1252 Microsoft (voir .../character-set-info/ windows-1252) [Wendt]

windows-1253 Microsoft (voir .../character-set-info/ windows-1253) [Lazhintseva]

windows-1254 Microsoft (voir .../character-set-info/ windows-1254) [Lazhintseva]

windows-1255 Microsoft (voir .../character-set-info/ windows-1255) [Lazhintseva]

windows-1256 Microsoft (voir .../character-set-info/ windows-1256) [Lazhintseva]

windows-1257 Microsoft (voir .../character-set-info/ windows-1257) [Lazhintseva]

windows-1258 Microsoft (voir .../character-set-info/ windows-1258) [Lazhintseva]

TIS-620 Institut thaïlandais des normes industrielles

(TISI) [Tantsetthi]

HZ-GB-2312 RFC 1842, RFC 1843 [RFC1842,

[RFC1843]

www.it-ebooks.info

Symboles

Indice

Nous aimerions connaître vos suggestions pour améliorer nos index. Envoyez-nous un e-mail à index@oreilly.com .

À propos des auteurs

David Gourley est directeur technique d'Endeca, où il dirige la recherche et le développement des produits. Endeca développe des solutions d'accès à l'information sur Internet et intranet qui offrent de nouvelles façons de naviguer et d'explorer les données d'entreprise. Avant de rejoindre Endeca, David était membre de l'équipe d'ingénierie fondatrice d'Inktomi, où il a contribué au développement de la base de données de recherche Internet et a joué un rôle clé dans le développement des produits de mise en cache web d'Inktomi.

David a obtenu une licence en informatique de l'Université de Californie à

Berkeley, et il détient plusieurs brevets dans les technologies Web.

Brian Totty était récemment vice-président de la R&D chez Inktomi Corporation (une entreprise qu'il a contribué à fonder en 1996), où il dirigeait la recherche et le développement de technologies de mise en cache web, de streaming multimédia et de recherche sur Internet. Auparavant, il était scientifique chez Silicon Graphics, où il concevait et optimisait des logiciels pour des systèmes de réseaux et de supercalcul hautes performances. Auparavant, il occupait un poste d'ingénieur au sein du groupe Technologies avancées d'Apple Computer.

Brian est titulaire d'un doctorat en informatique de l'Université de l'Illinois à Urbana-Champaign et d'une licence en informatique et génie électrique du MIT, où il a reçu le prix Organick pour ses recherches en systèmes informatiques. Il a également développé et enseigné des cours primés sur les technologies Internet pour le compte de l'université.

Système d'extension de l'Université de Californie.

Marjorie Sayer écrit sur les logiciels de mise en cache réseau chez Inktomi Corporation. Après avoir obtenu une maîtrise et un doctorat en mathématiques à l'Université de Californie à Berkeley, elle a travaillé sur la réforme des programmes de mathématiques. Depuis 1990, elle écrit sur la gestion des ressources énergétiques, les logiciels de systèmes parallèles, la téléphonie et les réseaux.

Sailu Reddy dirige actuellement le développement de proxys HTTP intégrés améliorant les performances chez Inktomi Corporation. Sailu développe des systèmes logiciels complexes depuis 12 ans et est fortement impliqué dans la recherche et le développement d'infrastructures web depuis 1995. Il a été l'un des principaux ingénieurs des premiers serveurs et proxys web de Netscape, ainsi que de plusieurs générations suivantes. Son expérience technique couvre les applications HTTP, les techniques de compression de données, les moteurs de bases de données et la gestion collaborative. Sailu est titulaire d'un master en systèmes d'information de l'université.

Université de l'Arizona et détient plusieurs brevets dans les technologies Web.

Anshu Aggarwal est directeur de l'ingénierie chez Inktomi Corporation. Il dirige les équipes d'ingénierie de traitement des protocoles pour les produits de mise en cache web d'Inktomi et participe à la conception des technologies web chez Inktomi depuis 1997. Anshu est titulaire d'une maîtrise et d'un doctorat en informatique de l'Université du Colorado à Boulder, spécialisé dans les techniques de cohérence mémoire pour les machines multiprocesseurs distribuées. Il est également titulaire d'une maîtrise et d'une licence en génie électrique.

Anshu est l'auteur de plusieurs articles techniques et détient deux brevets.

Colophon

Notre style est le fruit des commentaires de nos lecteurs, de nos propres expérimentations et des retours de nos réseaux de distribution. Des couvertures originales complètent notre approche unique des sujets techniques, insufflant personnalité et vie à des sujets potentiellement arides.

L'animal figurant sur la couverture de HTTP : The Definitive Guide est un spermophile à treize lignes (Spermophilus tridecemlineatus), commun dans le centre de l'Amérique du Nord. Fidèle à son nom, le spermophile à treize lignes arbore treize rayures ornées de rangées de points clairs sur toute la longueur de son dos. Son motif coloré se fond dans son environnement, le protégeant des prédateurs. Les spermophiles à treize lignes font partie de la famille des écureuils, qui comprend les tamias, les spermophiles, les écureuils arboricoles, les

chiens de prairie et les marmottes. Leur taille est similaire à celle du tamia rayé, mais plus petite que celle de l'écureuil gris commun, mesurant en moyenne environ 28 cm (queue de 12 à 15 cm comprise).

Les spermophiles rayés hibernent en octobre et émergent fin mars ou début avril. Chaque femelle donne généralement naissance à une portée de 7 à 10 petits chaque mois de mai. Les jeunes quittent leur terrier à l'âge de quatre à cinq semaines et atteignent leur taille adulte à six semaines. Les spermophiles préfèrent les zones ouvertes à l'herbe courte et aux sols sableux ou limoneux bien drainés pour leurs terriers. Ils évitent les zones boisées ; les pelouses tondues, les terrains de golf et les parcs sont des habitats courants.

Les écureuils terrestres peuvent causer des problèmes en creusant des terriers, en déterrant les graines fraîchement plantées et en endommageant les potagers. Cependant, ils constituent une proie importante pour plusieurs prédateurs, dont les blaireaux, les coyotes, les faucons, les belettes et divers serpents. Ils profitent également directement aux humains en se nourrissant de nombreuses mauvaises herbes, graines de mauvaises herbes et insectes nuisibles.

Rachel Wheeler a assuré la rédaction et la correction de HTTP : The Definitive Guide. Leanne Soylemez, Sarah Sherman et Mary Anne Weeks Mayo ont assuré le contrôle qualité, tandis que Derek Di Matteo et Brian Sawyer ont apporté leur aide à la production. John Bickelhaupt a rédigé l'index.

Ellie Volckhausen a conçu la couverture de ce livre, d'après une maquette d'Edie Freedman. L'image de couverture est une illustration originale de Lorrie LeJeune. Emma Colby a réalisé la mise en page de la couverture avec QuarkXPress 4.1 et la police ITC Garamond d'Adobe.

David Futato et Melanie Wang ont conçu l'aménagement intérieur, d'après une série de dessins de David Futato. Joe Wizda a préparé les fichiers pour la production dans FrameMaker 5.5.6. La police de texte est Linotype Birka ; la police de titre est Adobe Myriad Condensed ; et la police de code est TheSans Mono Condensed de LucasFont. Les illustrations du livre ont été réalisées par Robert Romano et Jessamyn Read avec Macromedia FreeHand 9 et Adobe Photoshop 6. Ce colophon a été rédigé par

Rachel Wheeler.