Groupe de travail Réseau
Request for Comments : 3229
Catégorie : En cours de normalisation
janvier 2002
Traduction Claude Brière de L'Isle

J. Mogul, Compaq WRL
B. Krishnamurthy, F. Douglis, AT&T
A. Feldmann, Univ. of Saarbruecken
Y. Goland, A. van Hoff, Marimba
D. Hellerstein, ERS/USDA

Codage des deltas dans HTTP

Résumé
Le présent document décrit comment le codage de deltas peut être pris en charge comme extension compatible à HTTP/1.1.

Beaucoup de demandes du protocole de transport hypertexte (HTTP, *Hypertext Transport Protocol*) causent la restitution d'instances légèrement modifiées de ressources dont le client a déjà une entrée en antémémoire. Les recherches ont montré que de telles mises à jour modificatrices sont fréquentes, et que les modifications sont normalement bien plus petites que l'entité réelle. Dans de tels cas, HTTP ferait une utilisation bien plus efficace de la bande passante du réseau si il pouvait transférer une description minimale des changements, plutôt que l'entière nouvelle instance de la ressource. C'est ce qu'on appelle le "codage des deltas".

Table des matières

1. Introduction

La Toile mondiale est un système réparti, et il bénéficie à ce titre de la mise en antémémoire pour réduire les délais de restitution. La restitution d'une ressource de la Toile (comme un document, une image, une icône, ou une appliquette) sur l'Internet ou autres réseaux de zone large prend habituellement assez de temps pour que le délai soit supérieur au seuil humain de perception. Souvent, ce délai se mesure en secondes. La mise en antémémoire peut souvent éliminer ou significativement réduire les délais de restitution.

De nombreuses ressources de la Toile changent avec le temps, de sorte qu'une approche pratique de la mise en antémémoire doit inclure un mécanisme de maintien de cohérence, pour éviter de présenter des informations périmées à l'utilisateur. À l'origine, le protocole de transfert hypertexte (HTTP, *Hypertext Transfer Protocol*) ne faisait que peu de place à la mise en antémémoire, mais sous la pression des événements, il a rapidement évolué vers une prise en charge d'un mécanisme simple de maintien de la cohérence des antémémoires.

Dans HTTP/1.0 [2], le serveur peut fournir un horodatage de la "dernière modification" avec une réponse. Si un client mémorise cette réponse dans une entrée d'antémémoire, et si ultérieurement il souhaite réutiliser la réponse, il peut transmettre un message de demande avec un champ "Si-modifié-depuis" qui contient cet horodatage ; c'est ce qu'on appelle une restitution conditionnelle *(conditional retrieval)*. À réception d'une demande conditionnelle, le serveur peut renvoyer une réponse complète, ou, si la ressource n'a pas changé, il peut envoyer une réponse abrégée, indiquant que l'entrée d'antémémoire du client est toujours valide. HTTP/1.0 comporte aussi un moyen pour que le serveur indique, via un horodatage "Expire", qu'une réponse ne sera valide que jusqu'à l'heure indiquée ; s'il en est ainsi, un client peut utiliser une copie mise en antémémoire de la réponse jusqu'au moment indiqué sans la valider d'abord en utilisant une restitution conditionnelle.

HTTP/1.1 [10] ajoute de nombreux dispositifs pour améliorer la cohérence et les performances des antémémoires. Cependant, il préserve le modèle tout ou rien pour les réponses aux restitutions conditionnelles : soit le serveur indique que la valeur de la ressource n'a pas changé du tout, soit il doit transmettre en entier la valeur actuelle.

Le bon sens suggère cependant (et les études le confirment) que même lorsque une ressource de la Toile change, la nouvelle instance est souvent en substance similaire à l'ancienne. Si la différence, ou le "delta", entre les deux instances pouvait être envoyée au client plutôt que la nouvelle instance entière, un client qui détient une copie en antémémoire de la vieille instance pourrait appliquer le delta pour construire la nouvelle version. Dans un monde où la bande passante est une grandeur finie, la réduction de taille et délai de réponse pourrait être significative.

On peut voir les deltas comme un moyen de tirer autant d'avantages que possible des antémémoires de clients et de mandataires. Plutôt que de traiter une réponse entière comme la "ligne d'antémémoire", avec les deltas on peut traiter des parties arbitraires d'une réponse en antémémoire comme unité remplaçable, et éviter de transférer des parties qui n'ont pas changé.

Le présent document propose un ensemble d'extensions compatibles à HTTP/1.1 qui permettent aux clients et serveurs d'utiliser le codage des deltas avec une redondance minimale.

On suppose que la spécification HTTP/1.1 est familière au lecteur.

1.1 Recherches et propositions

L'idée du codage des deltas pour réduire les coûts de communication ou de mémorisation n'est pas nouvelle. Par exemple, la norme de compression de la vidéo MPEG-1 transmet à l'occasion des trames d'image fixe, mais la plupart des trames envoyées sont codées (en simplifiant beaucoup) comme des changements à une trame adjacente. Les systèmes SCCS et RCS [27] pour le contrôle de la version de logiciel représentent les versions intermédiaires comme des deltas ; SCCS

commence avec une version d'origine et code les suivantes avec la transmission des deltas, tandis que RCS code les versions précédentes comme des deltas inversés à partir de leurs successeurs. La technique de Jacobson pour la compression des en-têtes IP et TCP sur des liaisons lentes [17] utilise une forme habile, très spécialisée, de codage des deltas.

En dépit de cette histoire, il apparaît qu'il a fallu plusieurs années avant que quiconque pense à appliquer le codage des deltas à HTTP, peut-être parce que le développement de la mise en antémémoire HTTP avait été fait un peu au hasard. La première suggestion publiée du codage des deltas apparaît dans un article de Williams et autres sur les politiques de suppression d'antémémoires HTTP [30], mais ces auteurs n'ont pas développé ce concept avant longtemps [29].

Le projet WebExpress [15] apparaît comme la première description publiée d'une mise en œuvre de codage des deltas pour HTTP (qu'ils appellent "différenciation"). WebExpress visait précisément les environnements sans fil, et comportait un certain nombre d'optimisations orthogonales. Aussi, le concept de WebExpress ne propose pas de changement au protocole HTTP lui-même, mais plutôt l'utilisation d'une paire de mandataires interposés pour convertir le flux de messages HTTP en une forme optimisée. Les résultats rapportés pour la différenciation WebExpress sont impressionnants, mais sont limités à quelques aspects choisis.

Banga et autres [1] décrivent l'utilisation de deltas optimisés, dans laquelle une couche de mandataires interposés sur l'une ou l'autre extrémité d'une liaison lente collaborent pour réduire la latence. Si le mandataire côté client a une copie en antémémoire d'une ressource, le mandataire côté serveur peut simplement envoyer un delta (ou une réponse 304 [Non Modifié]). Si le seul mandataire du côté serveur a une copie en antémémoire, il peut au mieux envoyer sa copie (éventuellement périmée) au mandataire côté client, suivie (si nécessaire) par un delta une fois que le mandataire côté serveur a validé sa propre entrée d'antémémoire avec le serveur d'origine. L'utilisation de deltas optimisés, à la différence du codage des deltas, augmente en fait le nombre d'octets envoyés sur le réseau, en tentant d'améliorer la latence en anticipant une réponse "Non Modifié" de la part du serveur d'origine. L'article sur les deltas optimisés, comme celui sur WebExpress, ne propose pas de changement au protocole HTTP lui-même, et les résultats ne rapportent qu'un petit ensemble d'URL choisis.

Mogul et autres [23] ont collecté de nombreux échantillons, sur deux sites différents, du contenu complet de messages HTTP, pour quantifier les avantages potentiels des réponses de codage de deltas. Ils ont montré que le codages des deltas peut fournir des améliorations remarquables en taille de réponse et en délai de réponse pour un sous-ensemble important de types de contenus HTTP. Ils proposaient un ensemble d'extensions HTTP, mais sans le niveau de détail requis pour une spécification. Douglis et autres [8] ont utilisé les mêmes ensembles d'échantillons de contenus complets pour quantifier le taux de changement des ressources sur la Toile.

Le protocole de distribution et de restitution (DRP, *Distribution and Replication Protocol*) proposé au W3C par Marimba, Netscape, Sun, Novell, et At Home, vise à fournir une collection de nouveaux dispositifs pour HTTP, pour prendre en charge "la duplication efficace de données sur HTTP" [13]. Un aspect de la proposition DRP est l'utilisation d'un "téléchargement différentiel," qui est essentiellement une forme de codage de deltas. La proposition DRP d'origine utilise une approche différente de celle décrite ici, mais une révision en cours de DRP le rendra conforme à la proposition du présent document.

Tridgell et Mackerras [28] décrivent l'algorithme "rsync", qui réalise quelque chose de similaire au codage des deltas. Dans rsync, le client casse une série d'entrées d'antémémoire en une série de blocs de taille fixe, calcule une valeur de résumé pour chaque bloc, et envoie la série des valeurs des résumés au serveur au titre de sa demande. Le serveur d'origine fait le même calcul sur la base du bloc, et ne retourne que les blocs dont les valeurs de résumé diffèrent. On pense qu'il serait possible de prendre en charge rsync en utilisant le cadre de "manipulation d'instance" décrit plus loin dans le présent document, mais ceci n'a pas été approfondi dans tous ses détails.

2. Objectifs

Les objectifs de cette proposition sont :
1. De réduire la taille moyenne des réponses HTTP, améliorant par là la latence et l'utilisation du réseau.
2. D'éviter tout aller-retour supplémentaire sur le réseau.
3. De minimise la quantité de redondance par demande et par réponse.
4. De prendre en charge divers algorithmes et formats de codage.
5. D'interopérer avec HTTP/1.0 et HTTP/1.1.
6. D'être entièrement facultatif pour les clients, les mandataires, et les serveurs.
7. De permettre des mises en œuvre relativement simples.

Il ne figure pas dans les objectifs de :

- Réduire le nombre de demandes HTTP envoyées à un serveur d'origine.
- Réduire la taille de chaque message HTTP.
- Augmenter le taux de touche des antémémoires HTTP.
- Permettre des mises en œuvre excessivement simplistes de codage des deltas.
- Coder les deltas des messages de demande, ou des réponses à des méthodes autres que GET.

Rien dans la présente spécification n'empêche spécifiquement l'utilisation d'un codage des deltas pour le corps d'une demande PUT. Cependant, aucun mécanisme n'existe actuellement pour que le client découvre si le serveur peut interpréter de tels messages, et donc, on ne tentera pas de spécifier comment cela pourrait être utilisé.

3. Terminologie

HTTP/1.1 [10] définit les termes suivants :

ressource Un objet ou service de données du réseau qui peut être identifié par un URI, comme défini au paragraphe 3.2. Des ressources peuvent être disponibles dans de multiples représentations (par exemple, plusieurs langages, formats de données, taille, résolutions) ou varier d'autres façons.

entité Les informations transférées comme charge utile d'une demande ou réponse. Une entité consiste en méta informations sous la forme de champs d'en-tête d'entité et de contenu sous la forme d'un corps d'entité, comme décrit à la section 7.

variante Une ressource peut avoir une, ou plus d'une, représentation associée à tout moment. Chacune de ces représentations est appelée une "variante". L'utilisation du terme "variante" n'implique pas nécessairement que la ressource est soumise à une négociation de son contenu.

La définition du dictionnaire pour "entité" est "quelque chose qui a une existence séparée et distincte et une réalité objective ou conceptuelle " [21]. Malheureusement, la définition de "entité" dans HTTP/1.1 est similaire à celle utilisée dans MIME [12], fondée sur une fausse analogie entre MIME et HTTP.

Dans MIME, les messages de la messagerie électronique ont bien des existences distinctes et séparées. MIME définit une "entité" comme quelque chose qui "se réfère spécifiquement aux champs d'en-tête définis par MIME et contient soit un message, soit une partie du corps d'une entité multi parties".

Dans HTTP, cependant, un message de réponse à GET n'a pas une existence distincte et séparée. Il reflète plutôt l'état actuel d'une ressource (ou d'une variante, sous réserve d'un ensemble de contraintes). La spécification HTTP/1.1 n'a pas de terme pour décrire "la valeur qui serait retournée en réponse à une demande GET à l'instant présent pour la variante choisie de la ressource spécifiée". Cela conduit à une formulation étrange dans la spécification HTTP/1.1 aux endroits ou ce concept est nécessaire.

Pour exprimer ce concept, on définit un nouveau terme, à utiliser dans le présent document :

instance L'entité qui serait retournée dans une réponse d'état 200 à une demande GET, à l'instant présent, pour la variante choisie de la ressource spécifiée, avec l'application de zéro, un ou plusieurs codages de contenu, mais sans l'application d'aucune manipulation d'instance (voir ci-dessous) ou codages de transfert.

Il est pratique de penser à une étiquette d'entité, dans HTTP/1.1, comme étant associée à une instance, plutôt qu'à une entité. C'est-à-dire que pour une certaine ressource, deux messages de réponse différents peuvent inclure la même étiquette d'entité, mais deux instances différentes de la ressource ne devraient jamais être associées à la même (forte) étiquette d'entité.

On utilisera le terme "delta" de façon informelle dans le présent document pour signifier une réponse HTTP codée comme la différence entre deux instances.

Plus formellement, les codages de delta sont des membres d'une classe potentiellement plus large de transformations sur des instances, ce qui nous conduit à ce nouveau terme :

manipulation d'instance C'est une opération sur une ou plusieurs instances qui peuvent résulter en ce qu'une instance soit convoyée du serveur au client en plusieurs parties, ou en plus d'un message de réponse. Par exemple, un choix de gamme ou un codage de delta. Les manipulations d'instances sont de bout en bout, et impliquent souvent l'utilisation d'une antémémoire chez le client.

Pour des raisons que l'on précisera plus loin, il est pratique de voir une sélection de sous gamme comme une forme de manipulation d'instance. Dans certains contextes, la compression peut aussi être traitée comme une manipulation d'instance, plutôt que comme un codage de contenu ou un codage de transfert.

4. Séquence de génération de message HTTP

HTTP/1.1 prend en charge un certain nombre de transformations différentes sur le corps d'une valeur :

Codage de contenu Selon la spécification, "les valeurs de codage de contenu indiquent une transformation de codage qui a été ou peut être appliquée à une entité. Les codages de contenu sont principalement utilisés pour permettre à un document d'être compressé ou autrement transformé de façon utile sans perdre l'identité de son type de support sous-jacent et sans perte d'information. Fréquemment, l'entité est mémorisée en forme codée, transmise directement, et seulement décodée par le receveur". Les codages de contenu sont normalement des transformations de bout en bout; c'est-à-dire qu'une fois appliqués chez l'envoyeur, ils ne sont pas retirés sauf chez le receveur ultime. Un serveur intermédiaire peut appliquer un codage de contenu, dans les circonstances appropriées.

Codage de transfert Selon la spécification, "les valeurs de codage de transfert sont utilisées pour indiquer une transformation de codage qui a été, peut être, ou peut devoir être appliquée à un corps d'entité afin d'assurer un "transport sûr" à travers le réseau. Cela diffère d'un codage de contenu en ce que le codage de transfert est une propriété du message, non de l'entité d'origine". Les codages de transfert sont explicitement des transformations bond par bond (bien que, comme optimisation, un mandataire intermédiaire puisse mémoriser la version en codage de transfert d'un message si ce comportement est cohérent avec sa fonction visible de l'extérieur).

Gammes Un client HTTP, utilisant l'en-tête Gamme, peut demander que le serveur retourne une ou plusieurs sous gammes de l'instance, plutôt que la valeur de l'instance entière. HTTP/1.1 ne prend en charge que les gammes d'octets, bien qu'il y ait quelque possibilité que de futures extensions permettent d'autres sortes de spécificateurs de gamme (comme les chapitres d'un document).

Un client signale sa volonté de recevoir un codage de contenu par l'envoi d'un en-tête "Codages-acceptés" qui fait la liste de l'ensemble des codages de contenu qu'il comprend. Il peut facultativement inclure des informations sur les codages de contenu qu'il préfère. Si un serveur utilise un ou des codages de contenu non identité, il inclut un champ d'en-tête "Codage-de-contenu" dans la réponse, qui fait la liste de ces codages de contenu dans leur ordre d'application.

La RFC2068 [9] ne comportait pas de mécanisme analogue pour négocier l'utilisation des codages de transfert, mais elle comportait un en-tête analogue "Codage-de-transfert" pour marquer la réponse. Un nouvel en-tête "TE" a été ajouté depuis à HTTP/1.1 [10], analogue à l'en-tête "Codages-acceptés".

Dans le présent document, on ajoute de nouveaux en-têtes facultatifs de message pour prendre en charge l'utilisation des manipulations d'instance. Un client signale sa volonté de recevoir une manipulation d'instance par l'envoi d'un en-tête "A-IM" (abrégé de "Accepte-Manipulation-Instance", qui est bien trop long à écrire) analogue à l'en-tête "Codages-acceptés". De la même façon, un serveur fait la liste de l'ensemble des manipulations d'instance qu'il a appliquées en se servant d'un en-tête "IM".

On doit comprendre les relations entre ces transformations afin de voir comment s'applique le codage de delta aux réponses HTTP.

Conceptuellement, les diverses transformations sont appliquées selon la séquence suivante :

1. À réception d'une demande GET, le serveur utilise l'URI qui est dans la demande pour identifier la ressource demandée.

2. Facultativement, il utilise les informations de la demande (et peut-être des informations supplémentaires) pour choisir une variante de cette ressource.

3. À ce moment, le serveur peut appliquer un codage de contenu non identité à l'instance, ou il peut y en avoir un qui est inhérent à sa génération. Cela résulte aussi en un en-tête Codage de contenu.

4. Le résultat de ces trois premières étapes, au moment où la demande est traitée, est une instance. L'instance comporte un

corps (éventuellement vide) et éventuellement des en-têtes d'instance. L'étiquette d'entité, s'il en est, est allouée à ce moment. C'est à dire qu'une étiquette d'entité est associée à une instance, PAS à une entité.

5. Le serveur peut alors appliquer une manipulation d'instance. Par exemple, si la demande incluait un en-tête Gamme *(Range)* le serveur peut facultativement produire une réponse de gamme, consistant en l'ensemble original d'en-têtes, l'en-tête Gamme de contenu *(Content-Range)* et la ou les gammes appropriées du corps (éventuellement codées). Les codages de delta sont des manipulation d'instances, et sont calculées à cette étape.

6. Le résultat de cette cinquième étape devient l'entité, consistant en en-têtes d'entité et en un corps d'entité.

7. Le serveur peut alors appliquer un codage de transfert non identité ; la compression au vol pourrait être effectuée à ce stade. S'il en est ainsi, un en-tête Codage de transfert sera ajouté au message.

8. Le résultat de la septième étape est le message, consistant en un corps de message (la version en codage de transfert du corps d'entité) en en-têtes d'entité, et en en-têtes supplémentaires de réponse et généraux.

Note : Le paragraphe 14.13 de la spécification de HTTP/1.1 [10] dit que "le champ d'en-tête d'entité Longueur du contenu indique la taille du corps de l'entité". En d'autres termes, Longueur de contenu mesure la longueur d'une entité, non d'une instance ou d'une variante. Par exemple, si le message est un codage de delta, Longueur de contenu donne la longueur du codage de delta, non la longueur de l'instance actuelle.

Voici un diagramme de la séquence :

Type de données	**opération conduisant au prochain type de données**
ressource	
↓	choisir une variante acceptable, si nécessaire,
variante	
↓	appliquer le codage de contenu, s'il en est,
↓	calculer/allouer une étiquette d'entité
instance	
↓	appliquer la manipulation d'instance, s'il en est, (codage de delta, choix de gamme, etc.)
corps d'entité	
↓	applique le codage de transfert, s'il en est un.
corps de message	

Cette formalisation de la séquence de génération du message HTTP n'avait pas été décrite auparavant. Cependant, il est clair que le choix de gamme doit être fait après que l'étiquette d'entité a été allouée et après l'application de tout codage de contenu, et avant l'application de tout codage de transfert. Donc, cette formalisation est parfaitement cohérente avec les pratiques et spécifications précédentes.

4.1 Relations entre deltas et gammes

Si les gammes et les codages de deltas sont tous deux des formes de manipulation d'instance, laquelle devrait être appliquée d'abord ? Cela dépend de la façon dont la gamme est utilisée.

Les gammes sont utilisées pour deux objets principaux, à la discrétion du client demandeur :
1. pour achever une réponse partielle après une terminaison prématurée d'une transmission de message,
2. pour obtenir les sections d'une instance qui viennent d'être choisies.

Dans la première utilisation de Gamme, il faudrait l'appliquer après tout codage de delta, car l'utilisation prévue est de récupérer une copie intacte de l'instance codée en delta. Dans la seconde utilisation de Gamme, il faudrait l'appliquer avant tout codage de delta, parce qu'autrement les décalages spécifiés dans la demande Gamme n'aurait pas de signification (le client ne peut généralement pas savoir comment le codage de delta d'un serveur transpose les décalages d'octet de l'instance en décalage d'octet de l'entité).

Nous avons donc besoin d'un mécanisme qui permette au client de spécifier l'ordre dans lequel deux ou plus manipulations d'instance devraient être appliquées. Cela est aisément fourni au titre de la spécification de l'en-tête "A-IM" (voir au paragraphe 10.5.3) qui exige que le serveur applique les manipulations d'instance dans l'ordre de leur énumération dans l'en-tête "A-IM". On inclut aussi un littéral "gamme" dans l'ensemble des manipulations d'instance enregistrées, pour

permettre au client de spécifier (par son rangement par rapport aux autres manipulations d'instance) si le choix de gammes est fait avant ou après le codage de delta.

On a aussi besoin d'un mécanisme pour que le serveur indique dans quel ordre deux manipulations d'instance ou plus ont été appliquées ; cette partie de la spécification de l'en-tête "IM" (voir au paragraphe 10.5.2) où on s'aligne sur les mêmes pratiques qu'utilisées pour l'en-tête "Codage de contenu" : l'en-tête "IM" fait la liste des manipulations d'instance dans l'ordre dans lequel elles ont été appliquées (incluant, peut-être, le littéral "gamme" particulier).

Un problème similaire survient lorsque les gammes sont combinées avec la compression. Si le client utilise une gamme pour compléter une réponse partielle après une terminaison prématurée d'un message compressé, alors la gamme devra être appliquée après la compression. Cela est faisable dans HTTP/1.1 non modifié, parce que la compression peut être faite comme un codage de contenu. Cependant, si le client utilise un Gamme pour obtenir des sections choisies d'une instance, il devrait normalement n'être capable de spécifier les décalages que selon les termes de la variante non compressée. Si la portion choisie est assez grande pour garantir la compression, le client pourrait demander un codage de transfert compressé, mais ceci est une transformation bond par bond et ce n'est pas l'approche la plus efficace (en particulier si un mandataire HTTP/1.0 se trouve sur le chemin).

On peut résoudre le problème en prenant en charge la compression comme manipulation d'instance (ainsi que comme un codage de contenu ou un codage de transfert) et en utilisant le nouveau mécanisme qui permet au client de spécifier que la manipulation d'instance Compression est faite après la manipulation d'instance Gamme.

Cela permet aussi au client de contrôler si la compression est faite avant ou après le codage de delta, car certains algorithmes simples de différenciation (comme la commande UNIX "diff") exigent la post-compression de leur résultat pour donner les meilleurs résultats.

5. Mécanismes de base

Dans cette section, on explique les concepts qui sous-tendent le codage de delta. Elle ne constitue pas la spécification formelle des extensions proposées ; voir cela à la section 10.

5.1 Fondements : vue d'ensemble de la validation d'antémémoire HTTP

Lorsque un client a une réponse dans son antémémoire, et souhaite s'assurer que cette entrée d'antémémoire est valide, HTTP/1.1 permet au client de faire un "GET conditionnel", en utilisant une des deux formes de "valideurs d'antémémoire". Dans la forme traditionnelle, disponible aussi bien dans HTTP/1.0 que dans HTTP/1.1, le client peut utiliser l'en-tête de demande "Si-Modifié-depuis" pour présenter au serveur l'horodatage "Dernière-modification" (s'il y en a un) que le serveur a fourni avec la réponse. Si l'horodatage du serveur pour la ressource n'a pas changé, il peut envoyer une réponse avec un code d'état de 304 (Non modifié) qui ne transmet pas le corps de la ressource. Si l'horodatage a changé, le serveur va normalement envoyer une réponse avec un code d'état de 200 (OK) qui porte une copie complète de la ressource, et un nouvel horodatage Dernière-modification.

Cette approche fondée sur l'horodatage est sujette à erreur à cause du manque de résolution de l'horodatage : si une ressource change deux fois en une seconde, le changement peut n'être pas détectable. Donc, HTTP/1.1 permet aussi au serveur de fournir une étiquette entité avec une réponse. Une étiquette entité est une chaîne opaque, construite par le serveur selon ses propres besoins ; la spécification du protocole impose un minimum simple d'exigences aux étiquettes d'entité. (En particulier, une étiquette d'entité "forte" doit changer si la valeur de la ressource change.) Dans ce cas, le client peut valider son entrée d'antémémoire en envoyant sa demande conditionnelle en utilisant l'en-tête de demande "Si-aucune-correspondance", présentant l'étiquette d'entité associée à la réponse en antémémoire. (Le protocole définit plusieurs autres façons de transmettre les étiquettes d'entité, comme l'en-tête "Si-Gamme", utilisé pour court-circuiter un aller-retour qui serait autrement nécessaire.) Si l'étiquette d'entité présentée correspond à l'étiquette actuelle du serveur pour la ressource, le serveur devrait envoyer une réponse 304 (Non modifié). Autrement, le serveur devrait envoyer une réponse 200 (OK) avec une copie complète de la ressource.

Dans le protocole HTTP existant (HTTP/1.0 ou HTTP/1.1) un client qui envoie une demande conditionnelle peut s'attendre à l'une de ces deux réponses :

- état = 200 (OK), avec une copie complète de la ressource, parce que la copie de la ressource du serveur est vraisemblablement différente de la copie en antémémoire du client.
- état = 304 (Non modifié), sans corps, parce que la copie de la ressource du serveur est vraisemblablement la même que la copie en antémémoire du client.

De façon informelle, on peut voir cela comme des "deltas" respectivement de 100 % et de 0 % de la ressource. Noter que ces deltas sont relatifs à une réponse en antémémoire spécifique. C'est-à-dire qu'un client ne peut pas demander un delta sans spécifier, d'une certaine manière, quelles sont les deux instances d'une ressource qui sont à différencier. La "nouvelle" instance est implicitement l'instance en cours que le serveur va retourner pour une demande inconditionnelle, et la "vieille" instance est celle qui est actuellement dans l'antémémoire du client. Le valideur d'antémémoire (heure de dernière modification ou étiquette entité) est ce qui est utilisé pour communiquer au serveur l'identité de la vieille instance.

5.2 Demande de transmission des deltas

Afin de prendre en charge la transmission des deltas réels, une extension à HTTP/1.1 doit fournir les dispositifs suivants :
1. Un moyen de marquer qu'une demande est conditionnelle.
2. Un moyen pour spécifier la vieille instance, à laquelle le delta sera appliqué par le client.
3. Un moyen pour indiquer que le client est capable d'appliquer une ou plusieurs formes spécifiques de codage de delta.
4. Un moyen de marquer qu'une réponse est codée en delta dans un certain format.

Les deux premiers dispositifs sont déjà fournis par HTTP/1.1 : la présence d'en en-tête de demande conditionnel (comme "Si-Modifié-depuis" ou "Si-Aucune-Correspondance") marque une demande comme conditionnelle, et la valeur de cet en-tête spécifie de façon univoque la vieille instance (ignorant le problème de la granularité du dernier horodatage modifié).

On reporte la discussion du dispositif 4 au paragraphe 5.6.

Le dispositif trois, un moyen pour le client d'indiquer qu'il est capable d'appliquer les deltas (à part la question triviale des deltas à 0 % et à 100 %) peut être réalisé en transmettant une liste de formats acceptables de codage en delta dans un champ d'en-tête de demande ; précisément, l'en-tête "A-IM". La présence de cette liste dans une demande conditionnelle indique que le client est capable d'appliquer des mises à jour d'antémémoire en codage de delta.

Par exemple, un client pourrait envoyer cette demande :

```
GET /foo.html HTTP/1.1
Host: bar.example.net
If-None-Match: "123xyz"
A-IM: vcdiff, diffe, gzip
```

La signification de cette demande est que :
- Le client veut obtenir la valeur actuelle de /foo.html.
- Il a déjà une réponse en antémémoire (instance) pour cette ressource, dont l'étiquette d'entité est "123xyz".
- Il est d'accord pour accepter les mises à jour par codage de delta en utilisant l'un des deux formats, "diffe" (c'est-à-dire, le résultat de la commande UNIX "diff -e") et "vcdiff". (Les algorithmes et formats de codage, tels que "vcdiff", sont décrits à la section 6.)
- Il est d'accord pour accepter des réponses qui ont été compressées en utilisant "gzip," qu'elles soient ou non en codage de delta. (Il peut être utile de compresser le résultat de "diff -e".) Cependant, sur la base des contraintes d'ordre obligatoire du paragraphe 10.5.3, si le codage de delta et la compression sont tous deux appliqués, cet en-tête de demande "A-IM" spécifie que la compression devrait être faite en dernier.

Si, dans cet exemple, l'étiquette d'entité actuelle du serveur pour la ressource est encore "123xyz", il devrait alors simplement retourner une réponse 304 (Non Modifié) comme le ferait un serveur traditionnel.

Si l'étiquette d'entité a changé, vraisemblablement mais pas nécessairement à cause d'une modification de la ressource, le serveur pourrait plutôt calculer le delta entre l'instance dont l'étiquette d'entité était "123xyz" et l'instance actuelle.

La discussion de ce que le serveur a besoin de mémoriser afin de calculer les deltas, se trouve à la section 7.

On note que si un client indique qu'il est d'accord pour accepter les deltas, mais que si le serveur ne prend pas en charge cette forme de manipulation d'instance, le serveur va simplement ignorer cet aspect de la demande. (HTTP permet toujours à une mise en œuvre d'ignorer un en-tête qui n'est pas exigé par une spécification à laquelle se conforme la mise en œuvre, et la spécification de "A-IM" permet au serveur d'ignorer une manipulation d'instance qu'il ne comprend pas.) Donc, si un serveur ne met pas en œuvre du tout l'en-tête A-IM, ou s'il ne met en œuvre aucune des manipulations d'instance énumérées dans l'en-tête A-IM, il agit comme si le client n'avait pas demandé une réponse en codage de delta : le serveur génère une réponse d'état 200.

5.3 Choix de l'algorithme et du format des deltas

Le serveur n'est pas obligé de transmettre une réponse codée en delta. Par exemple, le résultat peut être plus grand que la taille actuelle de la ressource. Le serveur pourrait n'être pas capable de calculer un delta pour ce type de ressource (par exemple, un format binaire compressé) ; le serveur pourrait n'avoir pas suffisamment de cycles de CPU pour le calcul du delta ; le serveur pourrait ne prendre en charge aucun des formats de delta acceptés par le client ; ou, la bande passante du réseau est tellement élevée que le délai impliqué par le calcul du delta ne vaut pas le délai gagné par l'envoi d'une plus petite réponse.

Cependant, si le serveur veut effectivement calculer un delta, et si l'ensemble de codages qu'il prend en charge a plus d'un codage en commun avec l'ensemble offert par le client, quel codage devrait il utiliser ? C'est essentiellement au choix du serveur, bien que le client puisse exprimer sa préférence en utilisant "Valeurs de qualité" (ou "qvalues") dans l'en-tête "A-IM". La spécification HTTP/1.1 [10] décrit les qvalues plus en détail. (Les clients peuvent préférer un format de codage de delta plutôt qu'un autre qui génère un plus petit codage, si les coûts de décodage pour le premier format sont inférieurs et si le client est pauvre en ressources.)

Les mises en œuvre de serveur ont un certain nombre d'approches possibles. Par exemple, si les cycles de CPU sont abondants et si la bande passante du réseau est restreinte, le serveur pourrait calculer chaque codage possible et ensuite envoyer le plus petit résultat. Ou le serveur pourrait utiliser une heuristique pour choisir un format de codage, sur la base de choses comme le type de contenu de la ressource, la taille actuelle de la ressource, et la quantité espérée de changements entre les instances de la ressource.

Noter qu'il peut être payant de mettre en antémémoire en interne les deltas chez le serveur, si une ressource est normalement demandée par plusieurs client différents qui ont la capacité de traiter les deltas entre les modifications. Dans ce cas, le coût de calcul d'un delta peut être amorti sur de nombreuses réponses, et ainsi le serveur peut utiliser un calcul plus coûteux.

5.4 Identification des réponses de deltas codés

Une réponse qui utilise le codage de delta doit être identifiée comme telle. Cela est fait en utilisant l'en-tête de réponse "IM", spécifié au paragraphe 10.5.2.

Cependant, une application simpliste de cette approche poserait de sérieux problèmes si une réponse codée de delta passe à travers une antémémoire intermédiaire (de mandataire) qui ne connaît pas le mécanisme de delta. Comme l'Internet comporte encore un nombre significatif d'antémémoires HTTP/1.0, qui pourraient n'être jamais entièrement remplacées, et parce que la spécification de HTTP insiste pour que les receveurs de message ignorent tout champ d'en-tête qu'ils ne comprennent pas, une antémémoire mandataire sans capacité de delta qui reçoit une réponse codée en delta pourrait mémoriser cette réponse, et pourrait ensuite la retourner à un client sans capacité de delta qui a fait une demande pour la même ressource. Ce client naïf croirait qu'il a reçu une copie valide de la ressource entière, avec des résultats désagréables prévisibles.

Pour résoudre ce problème, on propose que les réponses codées en delta (en fait, toute réponse avec une manipulation d'instance) soit identifiée comme telle en utilisant un nouveau code d'état HTTP. Pour être précis dans l'exposé qui suit, on utilisera le code (actuellement non alloué) de 226, avec une phrase de cause de "IM Utilisée". (On ne voit aucun avantage à épeler les mots "Manipulation d'instance utilisée", car cela exige la transmission d'octets inutiles, et que cette phrase de cause ne devrait normalement pas être vue par les utilisateurs humains.) Il y a quelques précédents à cette approche : la spécification HTTP/1.1 introduit le code d'état 206 (Contenu partiel) pour la transmission de sous gammes d'une ressource. Les mandataires existants transmettent apparemment des réponses avec des codes d'état inconnus, et ne tentent pas de les mettre en antémémoire.

Une solution de remplacement à l'utilisation d'un nouveau code d'état serait d'utiliser l'en-tête "Expires" pour empêcher les antémémoires HTTP/1.0 de mémoriser la réponse, puis d'utiliser "Cache-Control: max-age" (défini dans HTTP/1.1) pour permettre aux antémémoires plus modernes de mémoriser les réponses codées en delta. Cela ajouterait beaucoup d'octets aux en-têtes de réponse, et ainsi réduirait l'efficacité du codage de delta. Il n'est aussi pas entièrement clair que cette approche supprime toute mise en antémémoire par tous les mandataires HTTP/1.0.

Nous sommes réticents à définir un code d'état supplémentaire au titre de la prise en charge du codage de delta. Cependant, on ne voit aucun autre moyen efficace de rester compatible avec la base déployée de mises en œuvre d'antémémoires HTTP/1.0.

5.5 Garantir la sécurité de l'antémémoire

Bien qu'on ne connaisse aucune mise en œuvre de mandataire HTTP/1.1 qui tenterait de mettre en antémémoire une réponse avec un code d'état 2xx inconnu, la spécification HTTP/1.1 ne permet pas ce comportement si la réponse porte un champ d'en-tête Expires ou Cache-Control qui permet explicitement la mise en antémémoire. Cela poserait un problème lorsque une réponse 226 (IM Utilisé) porte de tels en-têtes.

La solution dans ce cas est d'exploiter le mécanisme Extensions de contrôle d'antémémoire de la spécification HTTP/1.1. On définit une nouvelle directive d'antémémoire, "im", qui indique que la directive d'antémémoire "no-store" peut être ignorée par les mises en œuvre qui se conforment à la spécification pour les en-têtes IM et A-IM.

Par exemple, cette réponse :

```
HTTP/1.1 226 IM Used
ETag: "489uhw"
IM: vcdiff
Date: Tue, 25 Nov 1997 18:30:05 GMT
Cache-Control: no-store, im, max-age=30

    ...
```

"NE DOIT PAS" être mémorisée par une antémémoire qui est conforme à la spécification HTTP/1.1 (qui déclare que la directive d'antémémoire max-age "implique que la réponse est mettable en antémémoire [...] sauf si une autre directive d'antémémoire, plus restrictive est aussi présente"). Cependant, une antémémoire qui ne se conforme pas à la spécification de la directive d'antémémoire im (c'est-à-dire, une antémémoire qui se conforme à la spécification pour les champs d'en-tête A-IM et IM, et le code d'état 226) ignore la directive no-store, et donc voit la directive max-age comme permettant la mise en antémémoire.

Nous ne sommes pas entièrement sûrs que toutes les antémémoires HTTP/1.1 obéissent à la règle que la directive max-age est outrepassée par la directive no-store. Si des essais de fonctionnement révèlent que cela pose un problème, des solutions plus élaborées sont possibles.

Avertissement aux mises en œuvre de serveur d'origine : il ne suffit pas d'envoyer

```
Vary: If-None-Match, A-IM
```

dans les réponses d'état 226. On a découvert au moins un scénario où cela n'empêche pas une antémémoire mandataire qui ne met pas en œuvre IM et A-IM de "valider" incorrectement une réponse 226 mise en antémémoire.

5.6 Transmission des réponses de deltas codés

Une réponse de delta codé diffère d'une réponse standard sous quatre aspects :
1. Elle porte un code d'état de 226 (IM Utilisé).
2. Elle porte un champ d'en-tête de réponse "IM" qui indique quel codage de delta est utilisé dans cette réponse.
3. Son corps de message est un codage de delta de l'instance en cours, plutôt qu'une copie complète de l'instance.
4. Elle peut porter plusieurs autres nouveaux en-têtes, comme décrit plus loin dans ce document.

Par exemple, une réponse à la demande donnée au paragraphe 5.2 pourrait ressembler à :

```
HTTP/1.1 226 IM Used
ETag: "489uhw"
IM: vcdiff
Date: Tue, 25 Nov 1997 18:30:05 GMT

    ...
```

(On ne montre pas le contenu réel du corps de réponse, car c'est une format binaire.)

Note : L'en-tête Etag dans une réponse 226 avec un codage de delta fournit l'étiquette d'entité de l'instance en cours de la variante de la ressource. Il n'y a pas de sens à associer une étiquette entité à la valeur de delta, qui n'est pas une instance.

5.7 Exemples de demandes combinant la gamme et le codage du delta

Dans l'exemple utilisé au paragraphe 5.2, le client envoie :

```
GET /foo.html HTTP/1.1
Host: bar.example.net
If-None-Match: "123xyz"
A-IM: vcdiff, diffe, gzip
```

et le serveur répond par une réponse 304 (Non modifié) ou par le codage de delta approprié.

Voici quelques autres exemples, pour préciser comment devraient être interprétée la demande du client.

Si le client envoie

```
GET /foo.html HTTP/1.1
Host: bar.example.net
If-None-Match: "123xyz"
A-IM: vcdiff, diffe, gzip, range
Range: bytes=0-99
```

la signification est alors la même que dans l'exemple ci-dessus, sauf qu'après le calcul du codage de delta (et de la compression, s'il en est une) le serveur retourne alors seulement les 100 premiers octets du résultat du codage de delta. (Si il est plus court que 100 octets, le codage de delta entier est retourné.) Parce que le jeton "range" apparaît en dernier dans l'en-tête "A-IM", cela dit au serveur d'origine d'appliquer tout choix de gamme après les autres manipulations d'instance.

L'interaction entre le mécanisme If-Range et le codage de delta est assez complexe. (If-Range signifie, en gros, "si l'entité est inchangée, envoyez moi la ou les parties qui me manquent ; autrement, envoyez moi la nouvelle entité entière.") Voici un exemple qui devrait éclaircir l'utilisation de cette combinaison.

Supposons que le client veuille avoir l'instance actuelle complète de http://bar.example.net/foo.html. Il a déjà une entrée d'antémémoire (complète) pour cet URI, avec l'étiquette d'entité "A", de sorte qu'il produit la demande suivante :

```
GET /foo.html HTTP/1.1
host: bar.example.net
If-None-Match: "A"
A-IM: vcdiff
```

Supposons que l'instance actuelle du serveur ait l'étiquette d'entité "B", et que le serveur ait aussi conservé une copie de l'instance avec l'étiquette entité "A". Alors, le serveur pourrait calculer la différence entre "B" et "A", et répondre par :

```
HTTP/1.1 226 IM Used
Etag: "B"
IM: vcdiff
Date: Tue, 25 Nov 1997 18:30:05 GMT
Content-Length: 1000

    ...
```

mais la connexion réseau s'est terminée après que le client a reçu exactement 900 octets du corps du message pour le contenu codé en delta.

Le client veut restituer les 100 octets restants du codage de delta qui était en cours d'envoi lors de l'interruption de la réponse. Il devrait donc envoyer :

```
GET /foo.html HTTP/1.1
host: bar.example.net
If-None-Match: "A"
If-Range: "B"
A-IM: vcdiff,range
Range: bytes=900-
```

Cette demande assez élaborée a une signification bien définie, qui dépend de l'étiquette d'entité actuelle Tcur de l'instance lorsque le serveur reçoit la demande :

Tcur = "A" (c'est-à-dire, pour certaines raisons, l'instance est revenue à la valeur qui est déjà dans l'antémémoire du client). Le serveur devrait retourner une réponse 304 (Non modifié) comme l'exige la spécification HTTP/1.1 pour "If-None-Match".

Tcur = "B" (c'est-à-dire, l'instance n'a pas changé de nouveau). La spécification HTTP/1.1 pour "If-None-Match", dans ce cas, est que le champ d'en-tête est ignoré (par un serveur qui ne comprend pas le codage de delta). Donc, cela est équivalent à la demande précédente du client, sauf que le choix de gamme est appliqué après la manipulation d'instance vcdiff (si toutes deux sont à appliquer). Donc, le serveur (capable de traiter le delta) calcule à nouveau le delta entre l'instance "A" et l'instance "B" (ou utilise un calcul mémorisé en antémémoire du delta) puis applique le choix de gamme, et retourne une réponse 226 (IM utilisé) avec un corps de message qui contient les octets 900 à 999 du résultat du codage vcdiff, avec un en-tête de réponse "IM:vcdiff,range".

Tcur = "C" (c'est-à-dire, l'instance a changé de nouveau). Dans ce cas, la spécification HTTP/1.1 pour "If-None-Match" signifie là encore que c'est équivalent à une demande inconditionnelle de l'instance actuelle. La spécification pour "If-Range" exige que le serveur retourne l'instance actuelle entière. Cependant, un serveur à capacité de delta peut construire le delta entre l'instance "A" décrite par le champ "If-None-Match" et l'instance ("C") actuelle, et retourner une réponse 226 (IM utilisé) avec un en-tête de réponse "IM:vcdiff".

Si la demande du client n'avait pas comporté le champ d'en-tête "If-None-Match: "A"", le serveur n'aurait pas pu calculer un delta, car il n'aurait pas su quelle instance entière était déjà disponible chez le client. Si la demande n'avait pas comporté le champ d'en-tête "If-Range: "B"", le serveur n'aurait pas pu faire la distinction entre les deux derniers cas (Tcur = "B" ou Tcur = "C") et n'aurait pas été capable d'appliquer le choix de gamme au résultat du codage de delta.

D'un autre côté, supposons que le client ait une entrée d'antémémoire pour l'instance "A" de http://bar.example.net/foo.html, et qu'il ait déjà reçu les 900 premiers octets d'une nouvelle instance "B" (peut-être par suite d'un transfert interrompu). Maintenant, le client veut recevoir l'entière instance actuelle, et il pourrait alors envoyer la demande suivante :

```
GET /foo.html HTTP/1.1
host: bar.example.net
If-None-Match: "A"
If-Range: "B"
A-IM: range,vcdiff
Range: bytes=900-
```

Dans cet exemple, comme dans l'exemple précédent, si Tcur = "A" alors le serveur devrait envoyer une réponse 304 (Non modifié) et si Tcur = "C", alors le serveur devrait envoyer l'entière nouvelle instance, soit comme réponse 200, soit comme un codage de delta par rapport à l'instance "A".

Cependant, si Tcur = "B", dans ce cas, le serveur devrait d'abord choisir la gamme spécifiée (de l'octet 900 à la fin) dans les deux instances "A" et "B", puis calculer le codage de delta entre ces gammes (en utilisant vcdiff) et ensuite transmettre le résultat en utilisant une réponse 226 (IM utilisé) avec un en-tête de réponse "IM:range,vcdiff".

6. Algorithmes et formats de codage

Un certain nombre d'algorithmes et formats de codage de delta ont été décrits dans la littérature :

diff –e Le programme UNIX "diff" est disponible partout, et il est relativement rapide aussi bien pour le codage que le décodage (en réalité, le décodages est fait à l'aide du programme "ed"). Cependant, la taille des deltas résultants est assez grande. Cet algorithme ne peut être utilisé que sur des fichiers en format texte.

diff -e | gzip Faire passer le résultat de "diff" par un algorithme de compression tel que "gzip" [5] (ou, peut-être encore mieux, "deflate" [7], [6]) donne un codage plus compact, mais le coût du codage et du décodage est bien plus élevé que pour "diff" tout seul. Cet algorithme ne peut être utilisé que sur des fichiers en format texte.

vcdiff (vdelta) L'algorithme qui génère le format "vcdiff" [19], [20] incorpore la compression de son résultat, et produit généralement de plus petits résultats que la combinaison de "diff" et de "gzip". L'algorithme tourne aussi

beaucoup plus vite, et peut s'appliquer à des entrées de format binaire. Le format "vcdiff" se fonde sur des travaux antérieurs sur un algorithme appelé "vdelta". (Noter que le format "vcdiff" peut être utilisé aussi bien pour le codage de delta que pour un format de compression, de sorte que deux différentes valeurs de manipulation d'instance devront être enregistrées afin de distinguer ces deux utilisations, si celle de format compressé devait être adoptée.) La plus récente étude publiée suggère que "vdelta" est le meilleur algorithme de delta global [16].

gdiff Le format gdiff [14] a été spécifié comme un format générique, indépendant des algorithmes, pour exprimer les deltas. Parce qu'il est plus générique, il est facile à mettre en œuvre, mais il peut n'être pas le format de codage le plus compact.

Notre proposition ne recommande pas un algorithme ou format spécifique, mais encourage plutôt les mises en œuvre de client et de serveur à choisir le ou les plus appropriés. Cependant, pour éviter la possibilité d'en-têtes "A-IM" excessivement longs, on suggère que, après une certaine période d'expérimentation, il soit raisonnable de spécifier un ensemble "recommandé" de formats de delta pour les mises en œuvre HTTP d'utilisation générale.

On peut imaginer qu'il serait possible de concevoir un algorithme de codage de delta approprié à un usage normal de codage d'image, comme GIF et JPEG. Bien que les expériences avec vdelta n'aient pas montré beaucoup de potentiel [23], cela peut être simplement parce que ces expériences utilisaient vdelta directement sur les formes déjà compressées de ces codages. Cependant, il pourrait être nécessaire d'imaginer un algorithme de codage de delta qui connaisse la nature bidimensionnelle des images. On peut s'attendre à ce que cela soit possible car la compression MPEG repose sur le calcul de deltas entre les trames successives d'un flux vidéo.

7. Gestion des instances de base

Si le délai entre des modifications d'une ressource est inférieur au délai normal de sortie des réponses dans les antémémoires de client, cela signifie que la "vieille instance" indiquée dans la demande conditionnelle d'un client pourrait ne pas se référer à la plus récente instance antérieure. Cela soulève la question du nombre de vieilles instances d'une ressource qui devraient être conservées par le serveur, si il en est. On appelle ces vieilles instances des "instances de base".

Il y a de nombreuses options possibles pour les mises en œuvre de serveur. Par exemple :
- Le serveur peut ne mémoriser aucune vieille instance, et ainsi ne va jamais répondre par un delta.
- Le serveur peut ne mémoriser que l'instance antérieure la plus récente ; les demandes qui tentent de valider cette instance pourront recevoir un delta en réponse, mais les demandes qui tentent de valider de plus vieilles instances recevront en réponse une copie complète de la ressource.
- Le serveur pourrait mémoriser toutes les instances antérieures, lui permettant de fournir une réponse de delta pour toute demande de client.
- Le serveur pourrait ne mémoriser qu'un sous ensemble des instances antérieures. L'utilisation de l'algorithme du moins récemment utilisé (LRU, *Least Recently Used*) pour déterminer cette sorte de sous ensemble s'est révélée efficace dans certaines circonstances similaires, comme un remplacement d'antémémoire.

Le serveur pourrait n'avoir pas à mémoriser explicitement les instances antérieures. Il pourrait à la place mémoriser juste les deltas entre des instances de base spécifiques et les instances suivantes (ou les deltas inverses entre les instances de base et les instances antérieures). Cette approche pourrait être intégrée à une antémémoire de deltas calculés.

Aucune de ces approches n'exige nécessairement une prise en charge de protocole supplémentaire. Cependant, si un administrateur de serveur veut ne mémoriser qu'un sous ensemble des instances antérieures, mais aimerait que le serveur soit capable de répondre en utilisant les deltas aussi souvent que possible, le client aura alors besoin de quelques informations supplémentaires. Autrement, l'"en-tête "If-None-Match" du client pourrait spécifier une instance de base non mémorisée chez le serveur, même si une instance de base appropriée est détenue dans l'antémémoire du client.

On a identifié deux changements supplémentaires au protocole pour aider à résoudre ce problème.

7.1 Plusieurs étiquettes d'entité dans l'en-tête If-None-Match

Bien que les exemples donnés jusqu'ici ne montrent qu'une seule étiquette d'entité dans un en-tête "If-None-Match", la spécification HTTP/1.1 permet que l'en-tête porte plus d'une étiquette d'entité. Ce dispositif a été introduit dans HTTP/1.1 pour prendre en charge une mise en antémémoire efficace de plusieurs variantes d'une ressource, mais il ne se restreint pas à cet usage.

Supposons qu'un client ait conservé plus d'une instance d'une ressource dans son antémémoire. C'est-à-dire que non seulement il a conservé l'instance la plus récente, mais il détient aussi des copies d'une ou plusieurs instances antérieures, invalides. (Autrement, il pourrait avoir conservé des deltas suffisants ou des informations de delta inverses pour reconstruire les plus anciennes instances.) Dans ce cas, il pourrait utiliser sa demande conditionnelle pour dire au serveur toutes les instances auxquelles il pourrait appliquer un delta. Par exemple, le client pourrait envoyer :

```
GET /foo.html HTTP/1.1
host: bar.example.net
If-None-Match: "123xyz", "337pey", "489uhw"
A-IM: vcdiff
```

pour indiquer qu'il a trois instances de cette ressource dans son antémémoire. Si le serveur est capable de générer un delta à partir d'une de ces instances antérieures, il peut choisir l'instance de base appropriée, calculer le delta, et retourner le résultat au client.

Dans ce cas, cependant, le serveur doit aussi dire au client quelle instance de base utiliser, et nous devons donc définir un en-tête de réponse, appelé "Delta-Base", à cette fin. Par exemple, le serveur pourrait répondre :

```
HTTP/1.1 226 IM Used
ETag: "1acl059"
IM: vcdiff
Delta-Base: "337pey"
Date: Tue, 25 Nov 1997 18:30:05 GMT
```

Cette réponse dit au client d'appliquer le delta à la réponse en antémémoire qui a l'étiquette d'entité "337pey", et d'associer l'étiquette d'entité "1acl059" au résultat.

Bien sûr, si le serveur a conservé plus d'une des instances antérieures identifiées par le client, cela pourrait compliquer le problème du choix du delta optimal à retourner, car maintenant, le serveur a un choix non seulement du format du delta, mais aussi de l'instance de base à utiliser.

7.2 Conseils pour la gestion de l'antémémoire client

La prise en charge de plusieurs étiquettes d'entité pour le choix de l'instance de base implique qu'un client pourrait bénéficier de la mémorisation de plusieurs instances antérieures d'une ressource dans son antémémoire. Un client avec un espace fini ne voudra pas conserver toutes les vieilles instances, de sorte qu'il doit gérer son antémémoire pour une efficacité maximale en sauvegardant seulement les instances qui ont le plus de chances d'être utiles pour des deltas futurs. Bien que cela puisse être réalisé en utilisant les informations purement locales au client (par exemple, un algorithme LRU) certaines informations de "conseil" provenant du serveur pourraient améliorer la capacité du client à gérer son antémémoire. L'utilisation de conseils pour améliorer les performances des antémémoires de la Toile a été décrite précédemment dans [4] et [22].

Si le serveur a l'intention de conserver certaines instances et pas d'autres, il peut étiqueter les réponses qui transmettent les instances conservées. Cela va aider le client à gérer son antémémoire, car il n'aura pas à conserver toutes les instances antérieures dans l'idée que seulement certaines d'entre elles pourraient être utiles plus tard. L'étiquette est un conseil au client, non une promesse que le serveur va conserver indéfiniment une instance.

On propose à cette fin d'ajouter une nouvelle directive à l'en-tête "Cache-Control" existant, appelé "retain". Par exemple, dans une réponse à une demande inconditionnelle, le serveur pourrait envoyer :

```
HTTP/1.1 200 OK
ETag: "337pey"
Date: Tue, 25 Nov 1997 18:30:05 GMT
Cache-Control: retain
```

pour suggérer qu'un client à capacité de delta devrait conserver cette instance. La directive "retain" pourrait aussi apparaître dans une réponse de delta, en se référant à l'instance en cours :

```
HTTP/1.1 226 IM Used
ETag: "1acl059"
Date: Tue, 25 Nov 1997 18:30:05 GMT
```

```
Cache-Control: retain
IM: vcdiff
Delta-Base: "337pey"
```

La directive "retain" comporte un paramètre de temporisation facultatif, que le serveur peut utiliser si il s'attend à supprimer une vieille instance de base à un instant donné. Par exemple,

```
HTTP/1.1 200 OK
ETag: "337pey"
Date: Tue, 25 Nov 1997 18:30:05 GMT
Cache-Control: retain=3600
```

signifie que le serveur a l'intention de conserver cette instance de base pendant une heure.

Une autre situation dans laquelle un serveur peut fournir un conseil à un client est lorsque le serveur prend en charge le mécanisme de delta en général, mais n'a pas l'intention de fournir des réponses de codages de delta pour une certaine ressource. En envoyant une directive "retain=0", il indique que le client ne devrait pas gâcher des octets d'en-tête de demande à tenter d'obtenir une réponse de codage de delta en utilisant cette instance de base (et, sous entendu, pour cette ressource). Il peut aussi indiquer que le client ne devrait pas gâcher de l'espace d'antémémoire sur cette instance après qu'elle est devenue périmée. Pour éviter de gaspiller des octets d'en-tête de réponse, un serveur devrait ne pas envoyer "retain=0", sauf en réponse à une demande qui tente d'obtenir une réponse de codage de delta.

Noter que la directive "retain" est orthogonale à la directive "max-age". La directive "max-age" indique pendant combien de temps une entrée d'antémémoire reste fraîche (c'est-à-dire, peut être utilisée sans contacter le serveur d'origine pour revalidation) ; la directive "retain" est intéressante pour un client APRÈS que l'entrée d'antémémoire est devenue périmée.

En pratique, le champ d'en-tête de réponse "Cache-Control" peut déjà être présent, de sorte que le coût (en octets) d'envoi de cette directive pourrait être plus faible que ce que ces exemples impliquent.

8. Deltas et antémémoires intermédiaires

Bien que nous ayons conçu les réponses en codage de delta de telle sorte qu'elles ne soient pas mémorisées par des antémémoires mandataires trop simplistes, si un mandataire comprend bien le mécanisme du delta, il pourrait être avantageux pour lui de participer à l'envoi et à la réception des deltas.

Un mandataire pourrait participer de plusieurs façons distinctes :

- En plus de transmettre une réponse codée en delta, le mandataire pourrait la mémoriser, puis l'utiliser pour répondre à une demande ultérieure avec un champ "If-None-Match" compatible (c'est-à-dire, qui est soit un sur ensemble du champ correspondant de la demande qui a provoqué la réponse la première fois, soit un qui comporte la valeur "Delta-Base" dans la réponse de l'antémémoire) et avec un champ d'en-tête de réponse "IM" compatible (qui comporte le format réel de codage de delta utilisé dans la réponse). Bien sûr, de telles utilisations sont soumises à toutes les autres règles de HTTP en ce qui concerne la validité des entrées d'antémémoire.

- En plus de la transmission d'une réponse codée en delta, le mandataire peut appliquer le delta à l'entrée appropriée dans sa propre antémémoire, qui pourrait alors être utilisée pour des réponses ultérieures (même de la part de clients sans capacité de delta).

- Lorsque le mandataire reçoit une demande conditionnelle de la part d'un client à capacité de delta, et que le mandataire a une copie complète d'une réponse mise à jour ("fraîche" dans la terminologie HTTP/1.1) dans son antémémoire, il pourrait générer localement un delta et le retourner au client demandeur.

- Lorsque le mandataire reçoit une demande de la part d'un client sans capacité de delta, il pourrait la convertir en une demande de delta avant de la transmettre au serveur, et ensuite (après avoir appliqué une réponse de delta résultante à une de ses propres entrées d'antémémoire) il retournerait une réponse de plein corps au client (ou une réponse avec le code d'état 206 ou 304, selon le cas approprié).

Toutes ces techniques facultatives augmentent la complexité des logiciels de mandataire, et pourraient augmenter les exigences de mémoire ou de CPU des mandataires. Cependant, appliqué avec discernement, cela devrait aider à réduire les latences vues par l'utilisateur final, et la charge du réseau. Généralement, la vitesse de CPU et les coûts de disque s'améliorent plus vite que les latences du réseau, de sorte qu'on espère voir une valeur croissante disponible de la part des

mises en œuvre de mandataires complexes.

9. Résumés pour l'intégrité des données

Lorsque un receveur ré-assemble une réponse HTTP complète à partir de plusieurs messages individuels, il peut être nécessaire de vérifier l'intégrité de la réponse complète. Par exemple, l'antémémoire du client pourrait être corrompue, ou la mise en œuvre du codage de delta (chez le client ou chez le serveur) pourrait avoir une bogue.

HTTP/1.1 comporte des mécanismes pour assurer l'intégrité des messages individuels. Un message peut inclure un en-tête de réponse "Content-MD5", qui fournit un résumé de message MD5 du corps du message (mais pas des en-têtes). Le mécanisme d'authentification par résumé de [11] fournit une fonction similaire de résumé de message, sauf qu'elle inclut les champs de certains en-têtes. Aucun de ces mécanismes n'a de disposition qui couvre un ensemble de données transmises sur plusieurs messages, comme ce serait le cas pour le résultat de l'application d'une réponse de codage de delta (ou, pour ce qui nous concerne, d'une réponse Gamme).

La protection de l'intégrité des données pour les messages ré-assemblés exige l'introduction d'un nouvel en-tête de message. Un tel mécanisme est proposé dans un autre document [24]. On peut vouloir quand même utiliser le mécanisme d'authentification par résumé, ou quelque chose de plus fort, pour protéger les messages de delta contre les altérations.

10. Spécification

Dans la présente spécification, les mots clés "DOIT", "NE DOIT PAS", "DEVRAIT", "NE DEVRAIT PAS", et "PEUT" sont à interpréter comme décrit dans la RFC2119 [3].

10.1 Spécifications des paramètres du protocole

La présente spécification définit un nouveau type de paramètre HTTP, une manipulation d'instance :

manipulation d'instance = jeton [imparams]
imparams = ";" imparam-name ["=" (jeton | chaîne entre guillemets)]
imparam-name = jeton

Noter que le imparam-name NE DOIT PAS être "q", pour éviter des ambiguïtés avec l'utilisation de qvalues (voir [10]).

L'ensemble des valeurs de manipulation d'instance est initialement :

- vcdiff
  Delta qui utilise le format de codage "vcdiff" [19], [20].

- diffe
  Résultat de la commande UNIX "diff -e" [26].

- gdiff
  Format de codage GDIFF [14].

- gzip
  Même définition que le codage de contenu HTTP "gzip".

- deflate
  Même définition que le codage de contenu HTTP "deflate".

- range
  Jeton qui indique que le résultat est un contenu partiel, car c'est le résultat d'un choix de gamme.

- identity
  Jeton qui n'est utilisé que dans l'en-tête A-IM (pas dans l'en-tête IM) pour indiquer si la manipulation d'instance identité est acceptable ou non.

Dans la suite de la présente spécification, on définit un sous ensemble de valeurs de manipulation d'instance comme valeurs de codage de delta :

codage de delta = "vcdiff" | "diffe" | "gdiff" | jeton

De futures valeurs de manipulation d'instance pourront aussi être incluses dans cette liste.

10.2 Considérations relatives à l'IANA

L'autorité d'allocation des numéros de l'Internet (IANA, *Internet Assigned Numbers Authority*) administre l'espace de noms des valeurs de manipulation d'instance. Les valeurs et leur signification doivent être documentées dans une RFC ou autre référence révisée par des pairs, permanente, et directement disponible, avec un degré de détail suffisant pour que soit possible l'interopérabilité entre des mises en œuvre indépendantes. Sous réserve de ces contraintes, les allocations de noms sont au premier qui les demande (voir la RFC2434 [25]).

La présente spécification insère aussi une nouvelle valeur dans le registre des codes d'état HTTP de l'IANA (voir la RFC2817 [18]). Voir au paragraphe 10.4.1 la spécification de ce code.

10.3 Exigences de base pour les réponses de codage de delta

Un serveur PEUT envoyer une réponse de codage de delta si toutes les conditions suivantes sont vérifiées :
1. le serveur devrait être capable d'envoyer une réponse 200 (OK) à la demande.
2. La demande du client comporte un champ d'en-tête A-IM qui fait la liste d'au moins un codage de delta.
3. La demande du client comporte un champ d'en-tête If-None-Match qui fait la liste d'au moins une étiquette d'entité valide pour une instance de l'URI de demande (une "instance de base").

Un réponse de codage de delta :
- DOIT porter un code d'état de 226 (IM utilisé).
- DOIT inclure un champ d'en-tête IM faisant la liste d'au moins, le codage de delta employé.
- PEUT inclure un champ d'en-tête Delta-Base qui fait la liste d'étiquette d'entité de l'instance de base.

10.4 Spécifications des codes d'état

Le nouveau code d'état suivant est défini pour HTTP.

10.4.1 226 IM Utilisé

Le serveur a satisfait à une demande GET pour la ressource, et la réponse est une représentation du résultat d'une ou plusieurs manipulations d'instance appliquée à l'instance actuelle. L'instance actuelle effective peut n'être pas disponible sauf en combinant cette réponse avec d'autres réponses antérieures ou futures, selon le cas approprié pour la ou les manipulations d'instance spécifiques. S'il en est ainsi, les en-têtes de l'instance résultante sont le résultat de la combinaison des en-têtes provenant de la réponse d'état 226 et des autres instances, suivant les règles du paragraphe 13.5.3 de la spécification HTTP/1.1 [10].

La demande DOIT comporter un en-tête A-IM énumérant au moins une manipulation d'instance. La réponse DOIT inclure un champ d'en-tête Etag qui donne l'étiquette d'entité de l'instance actuelle.

Une réponse reçue avec un code d'état de 226 PEUT être mémorisée par une antémémoire et utilisée dans une réponse à une demande ultérieure, sous réserve du mécanisme d'expiration HTTP et des en-têtes de contrôle d'antémémoire, et des exigences du paragraphe 10.6.

Une réponse reçue avec un code d'état de 226 PEUT être utilisée par une antémémoire, en conjonction avec une entrée d'antémémoire pour l'instance de base, pour créer une entrée d'antémémoire pour l'instance actuelle.

10.5 Spécifications de l'en-tête

Les en-têtes suivants sont définis pour être utilisés comme en-têtes d'entité. (Du fait de la confusion terminologique exposée à la section 3, certains en-têtes d'entité sont plus correctement associés à des instances qu'à des entités.)

10.5.1 Delta-Base

Le champ d'en-tête d'entité Delta-Base est utilisé dans une réponse codée en delta pour spécifier l'étiquette d'entité de l'instance de base.

Delta-Base = "Delta-Base" ":" étiquette d'entité

Un champ d'en-tête Delta-Base DOIT être inclus dans une réponse avec un en-tête IM qui comporte un codage de delta, si la demande comportait déjà plus d'une étiquette d'entité dans son champ d'en-tête If-None-Match.

Toute réponse avec un en-tête IM qui comporte un codage de delta PEUT inclure un en-tête Delta-Base.

Il n'est pas connu d'autre cas où une réponse de codage de delta DOIVE ou DEVRAIT inclure un en-tête Delta-Base, mais il n'y a pas eu d'analyse exhaustive ou formelle. Les mises en œuvre seraient bien avisées d'inclure un en-tête Delta-Base dans chaque réponse de codage de delta.

Une antémémoire ou un mandataire qui reçoit une réponse à codage de delta où manque un en-tête Delta-base PEUT ajouter un en-tête Delta-Base dont la valeur est l'étiquette d'entité donnée dans le champ If-None-Match de la demande (mais seulement si ce champ donne exactement une étiquette d'entité).

10.5.2 IM

Le champ d'en-tête de réponse IM est utilisé pour indiquer les manipulations d'instance, s'il en est, qui ont été appliquées à l'instance représentée par la réponse. Les manipulations d'instance normales incluent le codage de delta et la compression.

IM = "IM" ":" #(manipulation d'instance)

Les manipulations d'instance sont définies au paragraphe 10.1.

Comme cas particulier, si la manipulation d'instance comporte à la fois un choix de gamme et au moins une autre manipulation d'instance non identité, le champ d'en-tête IM DOIT être utilisé pour indiquer l'ordre dans lequel toutes ces manipulations d'instance, y compris le choix de gamme, ont été appliquées. Si l'en-tête IM cite la manipulation d'instance "gamme", la réponse DOIT inclure soit un en-tête Gamme de contenu, soit un multipartie/gamme d'octet Type de contenu dans lequel chaque partie contient un en-tête Gamme de contenu. (Voir au paragraphe 10.10 la discussion spécifique de la combinaison du codage de delta et de multipartie/gamme d'octet.)

Les réponses qui comportent un en-tête IM DOIVENT porter un code d'état de réponse de 226 (IM utilisé) comme spécifié au paragraphe 10.4.1.

Le serveur DEVRAIT omettre l'en-tête IM si il ne va citer que la manipulation d'instance "gamme". De telles réponses seront normalement envoyées avec un code d'état de réponse 206 (Contenu partiel) comme spécifié par HTTP/1.1 [10].

Voici un exemple d'utilisation de l'en-tête IM :

IM: vcdiff

Cet exemple indique que le corps d'entité est un codage de delta de l'instance, qui utilise le codage vcdiff.

IM: diffe, deflate, range

Cet exemple indique que l'instance a d'abord été codée en delta en utilisant le codage diffe, puis que le résultat a été compressé en utilisant deflate, et que finalement une ou plusieurs gammes de ce codage compressé ont été choisies.

IM: range, vcdiff

Cet exemple indique que un ou plusieurs gammes de l'instance ont été choisies, et que le résultat a été codé en delta par rapport à des gammes identiques d'une instance de base précédente.

Une antémémoire qui utilise une réponse reçue dans une réponse à une demande pour répondre à une demande ultérieure DOIT suivre les règles du paragraphe 10.6 si la réponse mise en antémémoire comporte un champ d'en-tête IM.

10.5.3 A-IM

Le champ d'en-tête de demande A-IM est similaire à Accept, mais restreint les manipulations d'instance (voir au paragraphe 10.1) qui sont acceptables dans la réponse. Comme spécifié au paragraphe 10.5.2, une réponse peut être le résultat de l'application de plusieurs manipulations d'instance.

A-IM = "A-IM" ":" #(manipulation d'instance [";" "q" "=" qvalue])

Lorsque un champ d'en-tête de demande A-IM inclut une ou plusieurs valeurs de codage en delta, la demande DOIT contenir un champ d'en-tête If-None-Match énumérant une ou plusieurs étiquettes d'entité provenant de réponses antérieures pour l'URI de demande.

Un serveur vérifie si une manipulation d'instance (parmi celles qu'il est capable d'employer) est acceptable, conformément à un certain champ d'en-tête A-IM, en utilisant les règles suivantes :

1. Si la manipulation d'instance figure dans la liste du champ A-IM, elle est alors acceptable, sauf si elle est accompagnée d'une qvalue de 0. (Comme défini au paragraphe 3.9 de la spécification HTTP/1.1 [10], une qvalue de 0 signifie "non acceptable".) Un serveur NE DOIT PAS utiliser une manipulation d'instance non-identité pour une réponse sauf si la manipulation d'instance figure dans la liste d'un en-tête A-IM dans la demande.

2. Si plusieurs manipulations d'instance sont acceptables mais incompatibles, la manipulation d'instance acceptable avec la plus forte qvalue différente de zéro est préférée.

3. La manipulation d'instance "identité" est toujours acceptable, sauf si spécifiquement refusée parce que le champ A-IM comporte "identity;q=0".

Si un champ A-IM est présent dans une demande, et si le serveur ne peut pas envoyer une réponse qui soit acceptable selon l'en-tête A-IM, alors le serveur DEVRAIT envoyer une réponse d'erreur avec le code d'état 406 (Non acceptable).

Si une réponse utilise plus d'une manipulation d'instance, les manipulations d'instance DOIVENT être appliquées dans l'ordre dans lequel elles apparaissent dans le champ d'en-tête de demande A-IM.

Le choix du serveur quant à appliquer ou non une manipulation d'instance DEVRAIT être indépendant de son choix d'appliquer toutes manipulations d'instance à deux entrées ultérieures à la réponse . (Les manipulations d'instance à deux entrées incluent les codages de delta, parce que ils prennent deux valeurs différentes en entrée. Les manipulations d'instance compression et "gamme" ne prennent qu'une seule entrée. D'autres manipulations d'instance pourront être définies à l'avenir.)

Note : L'intention de cette exigence est d'empêcher le serveur de générer une réponse codée en delta que le client ne pourra décoder qu'en appliquant d'abord un codage de manipulation d'instance à son instance de base en antémémoire. Une mise en œuvre de serveur peut souhaiter considérer ce que le client devrait logiquement avoir dans son antémémoire, lorsque elle décide des manipulations d'instance à appliquer avant un codage de delta.

Exemples :

A-IM: vcdiff, gdiff

Cet exemple signifie que le client acceptera un codage de delta dans l'un ou l'autre format vcdiff ou gdiff.

A-IM: vcdiff, gdiff;q=0.3

Cet exemple signifie que le client acceptera un codage de delta dans le format vcdiff ou gdiff, mais préfère le format vcdiff.

A-IM: vcdiff, diffe, gzip

Cet exemple signifie que le client acceptera un codage de delta dans l'un ou l'autre format vcdiff ou diffe, et acceptera le résultat du codage de delta compressé avec gzip. Il signifie aussi que le client acceptera une compression gzip de l'instance, sans aucun codage de delta, parce que A-IM ne donne aucun moyen d'insister pour que gzip ne soit utilisé que si diffe est

utilisé.

Le choix des combinaisons utiles de manipulations d'instance acceptables (par exemple, faire suivre diffe par gzip est utile, mais faire suivre vcdiff par gzip n'est probablement pas utile) appartient à la mise en œuvre de serveur.

10.6 Règles de mise en antémémoire pour les réponses 226

Lorsque un client ou mandataire reçoit une réponse 226 (IM utilisé) il PEUT utiliser cette réponse pour créer une entrée d'antémémoire de trois façons :

1. Il PEUT décoder toutes les manipulations d'instance pour récupérer l'instance d'origine, et mémoriser cette instance dans l'antémémoire. Dans ce cas, l'instance récupérée est mémorisée comme une réponse d'état 200, et DOIT être utilisée conformément aux règles normale de mise en antémémoire de HTTP.

2. Il PEUT décoder toutes les manipulations d'instance sauf pour la ou les choix de gamme, et mémoriser le résultat dans l'antémémoire. Dans ce cas, le résultat est mémorisée comme une réponse d'état 206, et DOIT être utilisé conformément au régime normal de mise en antémémoire de HTTP pour un contenu partiel.

3. Il PEUT mémoriser la réponse d'état 226 (IM utilisé) comme entrée d'antémémoire.

Une entrée d'antémémoire d'état 226 NE DOIT PAS être utilisée en réponse à une demande ultérieure dans une des conditions suivantes (une antémémoire qui ne mémorise jamais de réponse d'état 226 peut ignorer ces vérifications):

1. Si une des valeurs de manipulation d'instance provenant du champ d'en-tête IM dans la réponse en antémémoire n'apparaît pas dans le champ d'en-tête A-IM de la demande suivante. La comparaison entre les en-têtes est faite en utilisant une correspondance exacte sur chaque valeur de manipulation d'instance incluant toutes les valeurs imparams associées (voir au paragraphe 10.1).

2. Si l'ordre des valeurs de manipulation d'instance qui apparaissent dans le champ d'en-tête IM en antémémoire diffère de l'ordre de cet ensemble de manipulations d'instance dans le champ d'en-tête A-IM de la demande suivante.

3. Si la mise en œuvre d'antémémoire n'a pas connaissance, ou n'est pas au moins conditionnellement conforme à la spécification d'une des valeurs de manipulation d'instance dans le champ d'en-tête IM en antémémoire.

Note : Cette règle permet d'étendre l'ensemble des manipulations d'instance sans provoquer d'erreur chez les mises en œuvre déployées d'antémémoire. La spécification de nouvelles manipulations d'instances peut inclure des règles de mise en antémémoire supplémentaires pour améliorer les taux de touche d'antémémoire chez les mises en œuvre concernées.

4. Si une des valeurs de manipulation d'instance dans le champ d'en-tête IM en antémémoire est un codage de delta, et si l'entrée d'antémémoire comporte un champ d'en-tête Delta-Base, et si cette étiquette d'entité Delta-Base n'est pas une des étiquettes d'entité énumérées dans un champ d'en-tête If-None-Match de la demande suivante.

5. Si une des valeurs de manipulation d'instance dans le champ d'en-tête IM en antémémoire est un codage de delta, si l'entrée d'antémémoire ne comporte pas de champ d'en-tête Delta-Base, et si le champ d'en-tête If-None-Match de la demande qui a conduit à cette entrée d'antémémoire ne correspond pas au champ d'en-tête If-None-Match de la demande suivante.

Si le champ d'en-tête IM de la réponse en antémémoire comporte la manipulation d'instance "gamme", une entrée d'antémémoire d'état 226 NE DOIT PAS être utilisée alors en réponse à une demande suivante si la réponse en antémémoire est incohérente avec la ou les valeurs de champ d'en-tête Gamme dans la demande, comme ce serait le cas pour une réponse 206 (Contenu partiel) en antémémoire.

Note : On ne connaît aucune spécification formelle publiée existante pour décider si une réponse d'état 206 en antémémoire est cohérente avec une demande ultérieure. On pense que l'une ou l'autre de ces conditions est suffisante :

   1. Les gammes spécifiées dans les en-têtes de la demande qui ont conduit à la réponse en antémémoire sont les mêmes que celles spécifiées dans les en-têtes de la demande ultérieure.
   2. Les gammes spécifiées dans la réponse en antémémoire sont les mêmes que celles spécifiées dans les en-têtes de la demande suivante.

Il pourrait être nécessaire de pousser l'analyse un peu plus loin.

10.7 Règles pour les deltas en présence de codages de contenu

L'utilisation du codage de delta avec des instances de contenu codé ajoute une légère complexité. Lorsque un client (qui peut être un mandataire) a reçu une réponse codée en delta, l'une et/ou l'autre de ces nouvelles réponses et une réponse antérieure en antémémoire peuvent avoir des codages de contenu non-identité. On spécifie des règles pour le serveur et le client, pour empêcher des situations où le client est incapable de donner un sens à la réponse du serveur.

10.7.1 Règles pour générer des deltas en présence de codages de contenu

Lorsque un serveur génère une réponse codée en delta, la liste des codages de contenu qu'utilise le serveur (c'est-à-dire, la valeur du champ d'en-tête Codage de contenu de la réponse) DEVRAIT être un préfixe de la liste des codages de contenu que le serveur aurait utilisé si il n'avait pas généré un codage de delta.

Cette exigence permet à un client qui reçoit une réponse de codage de delta d'appliquer le delta à une instance de base en antémémoire sans avoir à appliquer de codage de contenu durant le processus (bien que le client puisse, bien sûr, être obligé de décoder certains codages de contenu).

10.7.2 Règles pour appliquer les deltas en présence de codage de contenu

Lorsque un client reçoit une réponse de delta avec un ou plusieurs codages de contenu non-identité :

1. Si la nouvelle réponse (delta) et la réponse en antémémoire (l'instance) ont toutes deux exactement le même ensemble de codages de contenu, le client applique la réponse de delta à la réponse en antémémoire sans retirer le codage de contenu de l'une ou l'autre réponse.

2. Si la nouvelle réponse (delta) et la réponse en antémémoire ont un ensemble différent de codages de contenu, avant d'applique le delta, le client décode un ou plusieurs codages de contenu de la réponse en antémémoire, jusqu'à ce que le résultat ait le même ensemble de codages de contenu que la réponse de delta.

3. Si un mandataire ou antémémoire transmet le résultat de l'application de la réponse de delta à une réponse d'instance de base en antémémoire, ou transmet ultérieurement ce résultat à partir d'une entrée d'antémémoire, la réponse transmise DOIT porter le même champ d'en-tête Codage de contenu que la nouvelle réponse (de delta) (et elle doit donc être à codage de contenu comme indiqué par ce champ d'en-tête).

L'intention de ces règles (et en particulier, de la règle n° 3) est que le résultat soit toujours cohérent avec la règle que l'étiquette d'entité soit associée au résultat du codage de contenu, et que tout receveur, après l'application du codage de delta reçoive exactement la même réponse que celle qu'il aurait reçu comme réponse d'état 200 de la part du serveur d'origine (sans aucun codage en delta).

10.7.3 Exemples pour utiliser A-IM, IM, et les codages de contenu

Supposons qu'un client, avec une antémémoire vide, envoie cette demande :

```
GET /foo.html HTTP/1.1
Host: example.com
Accept-encoding: gzip
```

et que le serveur d'origine réponde par :

```
HTTP/1.1 200 OK
Date: Wed, 24 Dec 1997 14:00:00 GMT
Etag: "abc"
Content-encoding: gzip
```

On utilisera la notation URI;étiquette d'entité pour noter les instances spécifiques, de sorte que cette réponse amènera le client à mémoriser dans son antémémoire entité GZIP(foo.html;"abc").

Supposons ensuite que le client, une minute plus tard, produise cette demande conditionnelle :

GET /foo.html HTTP/1.1
Host: example.com
If-none-match: "abc"
Accept-encoding: gzip
A-IM: vcdiff

Si le serveur est capable de générer une réponse en codage de delta, il pourrait choisir une de ces deux solutions. La première est de calculer le delta à partir de l'instance compressée (bien que cela puise ne pas donner le codage le plus efficace) :

HTTP/1.1 226 IM Used
Date: Wed, 24 Dec 1997 14:01:00 GMT
Etag: "def"
Delta-base: "abc"
Content-encoding: gzip
IM: vcdiff

Le corps de cette réponse serait le résultat de VCDIFF_DELTA(GZIP(foo.html;"abc"), GZIP(foo.html;"def")). Le client mémoriserait comme nouvelle entrée d'antémémoire l'entité GZIP(foo.html;"def"), après avoir récupéré cette entité en appliquant le delta à sa précédente entrée d'antémémoire.

L'autre solution du serveur serait de calculer le delta à partir des valeurs non compressées, en retournant :

HTTP/1.1 226 IM Used
Date: Wed, 24 Dec 1997 14:01:00 GMT
Delta-base: "abc"
Etag: "ghi"
IM: vcdiff

Le corps de cette réponse serait le résultat de VCDIFF_DELTA(GUNZIP(GZIP(foo.html;"abc")), foo.html;"ghi"), ou plus simplement VCDIFF_DELTA(foo.html;"abc", foo.html;"ghi"). Le client mémoriserait comme nouvelle entrée d'antémémoire l'entité foo.html;"ghi" (c'est-à-dire, sans aucun codage de contenu) après récupération de cette entité en appliquant le delta à sa précédente entrée d'antémémoire.

Noter que la nouvelle valeur de foo.html (à 14:01:00 GMT) sans le codage de contenu gzip doit avoir une étiquette d'entité différente de l'instance compressée du même fichier sous-jacent.

La seconde demande du client pourrait avoir été :

GET /foo.html HTTP/1.1
Host: example.com
If-none-match: "abc"
Accept-encoding: gzip
A-IM: diffe, gzip

Le client met gzip sur la liste dans les deux en-têtes Accept-Encoding et A-IM, parce que si le serveur ne prend pas en charge le codage de delta, le client aimerait au moins obtenir le bénéfice de la compression (comme codage de contenu). Cependant, si le serveur ne prend pas en charge le codage de delta diffe, le client aimerait que le résultat soit compressé, et cela doit être fait comme manipulation d'instance.

Un serveur qui prend en charge diffe pourrait répondre :

HTTP/1.1 226 IM Used
Date: Wed, 24 Dec 1997 14:01:00 GMT
Delta-base: "abc"
Etag: "ghi"
IM: diffe, gzip

Le corps de cette réponse serait le résultat de GZIP(DIFFE_DELTA(GUNZIP(GZIP(foo.html;"abc")), foo.html;"ghi")) ou plus simplement GZIP(DIFFE_DELTA(foo.html;"abc", foo.html;"ghi")). Comme la compression gzip est, dans ce cas, une manipulation d'instance et non un codage de contenu, elle n'est pas retenue lorsque la réponse réassemblée est mémorisée ou transmise, de sorte que le client va mémoriser comme nouvelle entrée d'antémémoire l'entité foo.html;"ghi" (sans aucun

codage de contenu ni compression).

10.8 Nouvelles directives de contrôle d'antémémoire

On définit deux nouvelles directives d'antémémoire (voir au paragraphe 14.9 de la RFC2616 [10] la spécification de directive d'antémémoire).

10.8.1 Directive Retain

L'ensemble des valeurs de directive de réponse d'antémémoire est augmenté pour inclure la directive retain.

cache-response-directive = ... "retain" ["=" delta-seconds]

Une directive retain est toujours un "conseil" de la part d'un serveur à un client ; elle ne spécifie jamais une action obligatoire pour le receveur.

La présence d'une directive retain indique qu'un client à capacité de delta devrait conserver l'instance contenue dans la réponse dans son antémémoire, si l'espace le permet, et devrait utiliser l'étiquette d'entité correspondante dans une demande future pour une réponse de codage de delta. C'est-à-dire que le serveur va vraisemblablement fournir des réponses codées en delta en utilisant l'instance correspondante comme instance de base. Cela implique que si un client a restitué et mis en antémémoire plusieurs instances d'une ressource, dont certaines sont marquées avec "retain" et certaines ne le sont pas, il n'est alors pas question de mettre en antémémoire les instances qui ne sont pas marquées "retain".

Si la directive retain comporte une valeur de delta de secondes, le serveur va alors vraisemblablement arrêter d'utiliser l'instance correspondante comme instance de base après le nombre de secondes spécifié. Un client ne devrait pas utiliser l'étiquette d'entité correspondante dans une future demande pour une réponse en codage de delta après la fin de ce délai. Le délai est mesuré depuis l'instant où la réponse est générée, de sorte qu'un client devrait inclure l'age de la réponse dans ses calculs.

Si la directive retain comporte une valeur de delta de secondes de zéro, un client NE DEVRAIT PAS utiliser l'étiquette d'entité correspondante dans une future demande pour une réponse en codage de delta.

Note : On recommande que les mises en œuvre de serveur considèrent les implications pour la bande passante de l'envoi d'une directive "retain=0" aux clients ou mandataires qui pourraient n'avoir pas la capacité d'en faire usage.

10.8.2 Directive IM

L'ensemble des valeurs de directive de réponse d'antémémoire est augmenté pour inclure la directive im.

cache-response-directive = ... | "im"

Une antémémoire qui se conforme à la spécification pour l'en-tête IMr, l'en-tête A-IMr, et le code d'état de réponse 226 DEVRAIT ignorer une directive d'antémémoire no-store si une directive im est présente dans la même réponse. Toutes les autres mises en œuvre DOIVENT ignorer la directive im (c'est-à-dire, DOIVENT observer une directive no-store, si elle est présente).

10.9 Utilisation de la compression avec le codage de delta

Il a été démontré que l'application de la compression de données aux codages de delta diffe et gdiff réduit considérablement la taille des corps de message résultants, dans de nombreux cas. (Le codage vcdiff est par ailleurs compressé par nature et ne bénéficie d'aucune autre compression.) Il est donc fortement recommandé que les mises en œuvre qui prennent en charge les codages de delta diffe et/ou gdiff prennent aussi en charge les codages de compression gzip et/ou deflate. (Le codage deflate fournit un résultat plus compact.) Cependant, ceci n'est pas exigé pour l'utilisation du codage de delta, principalement parce que les coûts de CPU associés à la compression et la décompression peuvent être excessifs dans certains environnements.

Un client qui prend en charge à la fois le codage de delta et la compression comme manipulations d'instance le signale par exemple, par

A-IM: diffe, deflate

La règle de rangement du paragraphe 10.5.3 exige que si le serveur utilise les deux manipulations d'instance dans la réponse, la compression soit appliquée au résultat du codage de delta, plutôt que l'inverse. C'est-à-dire que dans ce cas, la réponse comporterait :

IM: diffe, deflate

Noter qu'un client peut accepter la compression soit comme un codage de contenu, soit comme une manipulation d'instance. Par exemple :

Accept-Encoding: gzip
A-IM: gzip, gdiff

Dans cet exemple, le serveur peut appliquer la compression gzip, soit comme un codage de contenu, soit comme une manipulation d'instance, avant le codage de delta. Il faut se souvenir que l'étiquette d'entité est allouée après le codage de contenu mais avant une manipulation d'instance, de sorte que ce choix affecte bien la sémantique du codage de delta.

10.10 Codage de delta et multipartie/gammes d'octets

Un client peut demander plusieurs gammes d'octets non contiguës dans une seule demande. La réponse du serveur utilise le type de support "multipartie/gamme d'octet" (au paragraphe 19.2 de [10]) pour porter plusieurs gammes dans une réponse. Si une réponse multipartie/gamme d'octet est codée en delta (c'est-à-dire, utilise un codage de delta comme une manipulation d'instance) les en-têtes qui se rapportent au delta sont associés à la réponse entière, et non aux seules parties individuelles. (Cela parce que il n'y a qu'une seule instance de base et une seule instance en cours impliquée.) Une réponse codée en delta avec plusieurs gammes DOIT utiliser le même codage en delta pour toutes les gammes.

Si un serveur choisit d'utiliser un codage de delta pour une réponse multipartie/gamme d'octet, il DOIT générer une réponse en accord avec les règles suivantes :

Lorsque une réponse multipartie/gamme d'octet utilise un codage de delta avant une sélection de gamme, les champs d'en-tête A-IM et IM citent le codage de delta avant le littéral "range". (On se rappelle que c'est l'approche retenue pour obtenir une réponse partielle après une terminaison prématurée d'une transmission de message.) Le serveur génère d'abord une séquence d'octets représentant la différence (delta) entre l'instance de base et l'instance actuelle, puis choisit les gammes d'octets spécifiées, et transmet chacune de ces gammes dans une partie du type de support multipartie/gamme d'octet.

Lorsque une réponse multipartie/gamme d'octet utilise un codage de delta après un choix de gammes, les champs d'en-tête A-IM et IM citent le codage de delta après le littéral "range". (On se rappelle que c'est l'approche retenue pour obtenir une version mise à jour des seules sections choisies d'une instance.) Le serveur choisit d'abord les gammes spécifiées à partir de l'instance en cours, et choisit aussi les mêmes gammes spécifiées de l'instance de base. (Certaines de ces gammes choisies peuvent être la séquence vide, si l'instance n'est pas assez longue.) Le serveur génère alors les différences individuelles (deltas) entre les paires de gammes, et transmet chacune de ces différences dans une partie du type de support multipartie/gamme d'octet.

11. Quantifier la redondance du protocole

Les changements de protocole proposés augmentent légèrement la taille des en-têtes de message HTTP. Dans le cas le plus simple, une demande conditionnelle (c'est-à-dire, une demande pour un URI pour lequel le client a déjà une entrée d'antémémoire) inclurait un en-tête de plus, par exemple :

A-IM:vcdiff

Cela fait environ 13 octets supplémentaires. Une étude récente [23] rapporte des taille moyennes de demande à partir de deux traces différentes de 281 et 306 octets, de sorte que l'augmentation nette de la taille de demande serait entre 4 % et 5 %.

Comme un client doit avoir une entrée d'antémémoire existante à utiliser comme base pour une réponse de codage de delta, il ne va jamais envoyer un "A-IM: vcdiff" (ou citer d'autres formats de codage de delta) pour ses demandes inconditionnelles. La même étude a montré qu'au moins 46 % des échantillons de longueur de demande sont pour des URL qui n'ont pas été vus précédemment dans l'échantillon ; cela signifie que pas plus d'environ la moitié des demandes

normales de client pourraient être conditionnelles (et la fraction réelle sera vraisemblablement plus petite, étant donnée la taille finie des antémémoires réelles).

L'étude a aussi montré que 64 % des réponses dans un échantillon de longueurs étaient pour des types de contenu d'image (GIF et JPEG). Comme noté à la section 6, on ne connaît pas actuellement de format de codage de delta pour de tels types d'image. Sauf si un client prend en charge un tel format de codage de delta, on peut supposer qu'il ne va pas demander un delta lorsque il fait une demande conditionnelle pour des types de contenu d'image.

Le rapprochement de ces divers facteurs suggère que l'augmentation moyenne de la taille d'en-tête de demande serait bien inférieure à 5 %, et probablement en dessous de 1 %.

Les réponses codées en delta portent des en-têtes légèrement plus longs. Dans le cas le plus simple, une réponse porte un en-tête de plus, par exemple :

IM:vcdiff

Cela fait environ 11 octets. D'autres en-têtes (tels que "Delta-Base") peuvent aussi être inclus. Cependant, aucun de ces en-têtes supplémentaires ne sera inclus sauf dans des antémémoires où un codage de delta est effectivement employé, et l'envoyeur de la réponse peut éviter d'envoyer un codage de delta si il en résulte une augmentation nette de la taille de réponse. Donc, une réponse en codage de delta ne devrait jamais être plus grande qu'une réponse régulière pour la même demande.

Les simulations suggèrent que, lorsque le codage de delta est couronné de succès, il économise plusieurs milliers d'octets [23]. Donc, ajouter quelques douzaines d'octets aux en-têtes de réponse ne devrait presque jamais contrebalancer les économies réalisées sur la taille du corps de message.

Finalement, l'utilisation de la directive de contrôle d'antémémoire "retain" pourrait causer quelque redondance supplémentaire. Certaines heuristiques de serveur pourraient réussir à limiter l'utilisation de ces en-têtes à des situations où elles optimiseraient probablement de futures réponses. Aucun de ces en-têtes n'est nécessaire pour les pus simples utilisations de codage de delta.

12. Considérations pour la sécurité

On ne connaît aucun aspect du mécanisme de base du codage de delta qui affecte la sécurité existante pour le protocole HTTP/1.1.

13. Remerciements

Phong Vo a fourni de nombreux conseils pour le choix des algorithmes et formats de codage de delta. Issac Goldstand et Mike Dahlin ont formulé nombre de commentaires utiles sur la spécification. Dave Kristol a suggéré de nombreuses corrections rédactionnelles.

14. Considérations sur la propriété intellectuelle

Une revendication de droits de propriété intellectuelle a été notifiée à l'IETF à l'égard de tout ou partie de la spécification contenue dans le présent document. Pour d'autres informations, consulter la liste en ligne des revendications de droits.

L'IETF ne prend position sur la validité ou la portée d'aucun droit de propriété intellectuelle ou d'autres droits qui pourraient être revendiqués au titre de la mise en œuvre ou l'utilisation de la technologie décrite dans le présent document ou sur la mesure dans laquelle toute licence sur de tels droits pourrait être ou non disponible ; pas plus qu'elle ne prétend qu'elle ait fait aucun effort pour identifier de tels droits. Des informations sur les procédures de l'IETF au sujet des droits dans la documentation en cours de normalisation et en rapport avec les normes peuvent être trouvées dans le BCP-11. Des copies des revendications de droits peuvent être disponibles à la publication et toutes les assurances de licences peuvent être rendues disponibles, ou le résultat de tentatives d'obtention d'une licence ou permission générale pour l'utilisation de tels droits de propriété par les mises en œuvre ou utilisateurs de la présente spécification peuvent être obtenus auprès du secrétariat de l'IETF.

L'IETF invite toute partie intéressée à porter à son attention tous droits de reproduction, brevets ou applications de brevets,

ou autres droits de propriété qui pourraient couvrir une technologie qui pourrait être nécessaire pour mettre en pratique la présente norme. Prière d'adresser les information au Directeur Général de l'IETF.

15. Références

[1] Gaurav Banga, Fred Douglis, and Michael Rabinovich. "Optimistic Deltas for WWW Latency Reduction". Proc. 1997 USENIX Technical Conference, Anaheim, CA, janvier 1997, pp. 289-303.

[2] T. Berners-Lee, R. Fielding, H. Frystyk, "Protocole de transfert Hypertext -- HTTP/1.0", RFC1945, mai 1996. *(Information)*

[3] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", RFC2119, BCP 14, mars 1997.

[4] Edith Cohen, Balachander Krishnamurthy, and Jennifer Rexford. "Improving End-to-End Performance of the Web Using Server Volumes and Proxy Filters". Proc. SIGCOMM '98, septembre 1998, pp. 241-253.

[5] P. Deutsch, "Spécification du format de fichier GZIP version 4.3", RFC1952, mai 1996. *(Information)*

[6] P. Deutsch, "Spécification du format DEFLATE de données compressées, version 1.3", RFC1951, mai 1996.

[7] P. Deutsch et J-L Gailly, "Spécification du format ZLIB de données compressées, version 3.3", RFC1950, mai 1996.

[8] Fred Douglis, Anja Feldmann, Balachander Krishnamurthy, and Jeffrey Mogul. "Rate of Change and Other Metrics: a Live Study of the World Wide Web". Proc. Symposium on Internet Technologies and Systems, USENIX, Monterey, CA, décembre 1997, pp. 147-158.

[9] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, "Protocole de transfert Hypertext -- HTTP/1.1", RFC2068, janvier 1997. (*Obsolète, voir* RFC2616) *(P.S.)*

[10] R. Fielding et autres, "Protocole de transfert hypertexte -- HTTP/1.1", RFC2616, juin 1999. *(D.S., MàJ par 2817)*

[11] J. Franks et autres, "Authentification HTTP : Authentification d'accès de base et par résumé", RFC2617, juin 1999.

[12] N. Freed et N. Borenstein, "Extensions de messagerie Internet multi-objets (MIME) Partie 1 : Format des corps de message Internet", RFC2045, novembre 1996. *(D. S., MàJ par 2184, 2231, 5335.)*

[13] Arthur van Hoff, John Giannandrea, Mark Hapner, Steve Carter, and Milo Medin. "The HTTP Distribution and Replication Protocol". Technical Report NOTE-DRP, World Wide Web Consortium, août 1997.

[14] Arthur van Hoff and Jonathan Payne. "Generic Diff Format Specification". Technical Report NOTE-GDIFF, World Wide Web Consortium, août 1997.

[15] Barron C. Housel and David B. Lindquist. "WebExpress: A System for Optimizing Web Browsing in a Wireless Environment". Proc. 2nd Annual Intl. Conf. on Mobile Computing and Networking, ACM, Rye, New York, novembre 1996, pp. 108-116.

[16] James J. Hunt, Kiem-Phong Vo, and Walter F. Tichy. "An Empirical Study of Delta Algorithms". IEEE Soft. Config. and Maint. Workshop, 1996.

[17] V. Jacobson, "Compression des en-têtes TCP/IP pour les liaisons série à faible débit", RFC1144, février 1990.

[18] R. Khare, S. Lawrence, "Mise à niveau de TLS au sein de HTTP/1.1", RFC2817, mai 2000. *(P.S.)*

[19] David G. Korn and Kiem-Phong Vo. "A Generic Differencing and Compression Data Format". Technical Report HA1630000-021899-02TM, AT&T Labs - Research, février 1999.

[20] D. Korn et autres, "Format générique de différenciation et de compression de données VCDIFF", RFC3284 ,juin 2002.

[21] Merriam-Webster. "Webster's Seventh New Collegiate Dictionary". G. & C. Merriam Co., Springfield, MA, 1963.

[22] Jeffrey C. Mogul. "Hinted caching in the Web". Proc. Seventh ACM SIGOPS European Workshop, Connemara,

Ireland, septembre 1996, pp. 103-108.

[23] Jeffrey C. Mogul, Fred Douglis, Anja Feldmann, and Balachander Krishnamurthy. "Potential benefits of delta encoding and data compression for HTTP". Research Report 97/4, DECWRL, juillet 1997.

[24] J. Mogul, A. Van Hoff, "Résumés d'instances dans HTTP", RFC3230, janvier 2002. *(P.S.)*

[25] T. Narten et H. Alvestrand, "Lignes directrices pour la rédaction d'une section Considérations relatives à l'IANA dans les RFC", RFC2434, BCP 26, octobre, 1998. *(Rendue obsolète par la RFC 5226)*

[26] The Open Group. "The Single UNIX Specification, Version 2 - 6 Vol Set for UNIX 98". Document number T912, The Open Group, février 1997.

[27] W. Tichy. "RCS - A System For Version Control". Software - Practice and Experience 15, 7 (juillet 1985) 637-654.

[28] Andrew Tridgell and Paul Mackerras. "The rsync algorithm". Technical Report TR-CS-96-05, Department of Computer Science, Australian National University, juin 1996.

[29] Stephen Williams. Communication personelle. http://ei.cs.vt.edu/~williams/DIFF/prelim.html.

[30] Stephen Williams, Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, and Edward A. Fox. "Removal Policies in Network Caches for World-Wide Web Documents". Proc. SIGCOMM '96, Stanford, CA, août 1996, pp. 293-305.

16. Adresse des auteurs

Jeffrey C. Mogul
Western Research Laboratory
Compaq Computer Corporation
250 University Avenue
Palo Alto, California, 94305, U.S.A.
téléphone : 1 650 617 3304
mél : JeffMogul@acm.org

Balachander Krishnamurthy
AT&T Labs - Research
180 Park Ave, Room D-229
Florham Park, NJ 07932-0971,
USA
mél : bala@research.att.com

Fred Douglis
AT&T Labs - Research
180 Park Ave, Room B-137
Florham Park, NJ 07932-0971,
USA
téléphone : 1 973 360-8775
mél : douglis@research.att.com

Anja Feldmann
University of Saarbruecken,
Computer Science Department
Im Stadtwald, Geb. 36.1, Zimmer 310
D-66123 Saarbruecken, Germany
mél : anja@cs.uni-sb.de

Arthur van Hoff
Marimba, Inc.
440 Clyde Avenue
Mountain View, CA 94043,
téléphone : 1 650 930 5283
mél : avh@marimba.com

Daniel M. Hellerstein
Economic Research Service, USDA
1909 Franwall Ave,
Wheaton MD 20902
tél. : 1 202 694-5613
mél : danielh@crosslink.net

Yaron Y. Goland
mél : yaron@goland.org

17. Déclaration complète de droits de reproduction

Remerciement

Le financement de la fonction d’édition des RFC est actuellement fourni par l’Internet Society.