IUP3 **7 Janvier 2003**

Examen du cours “Internet”

Durée : 2h.
Documents autorisés : tous.
Note : Toutes les réponses doivent être clairement justifiées. Le barème est indicatif.

Nous rappelons les notions utilisées dans le mécanisme de *négociation de contenu* du RFC 2616 (HTTP/1.1) :

ressource : une entité identifiée par un URI.

représentation : une suite d'octets ayant un nom dérivé du nom de la ressource et un type, une langue, un encodage et un ensemble de caractères définis. Une ressource peut avoir plusieurs représentations.

ressource négociable : une ressource ayant plusieurs représentations.

variante : une représentation d'une ressource.

dimension de négociation : les manières de variation des représentations (variantes) pour une ressource négociable. Dans HTTP/1.1, ces dimensions sont : le type, la langue, l'encodage et l'ensemble de caractères utilisé.

1. HTTP (5 points)

1. Expliquer l'échange client-serveur dans la négociation de contenu.
2. Indiquer comment le client spécifie les dimensions de la négociation.
3. Expliquer pour l'en-tête de requête suivant les préférences exprimées :

   `Accept: text/html, image/gif;q=0.8, text/plain;q=0.5, */*;q=0.01`
4. Donner les trois réponses possibles du serveur dans la négociation de contenu, en indiquant quels doivent être l'en-tête et le corps de chaque réponse.
5. Quel est le problème que pose la négociation de contenu pour “cacher” les réponses du serveur ?

2. Apache (12 points)

Apache implémente le mécanisme de négociation de contenu. Pour une ressource, il peut envoyer sa meilleure représentation en fonction des préférences transmises par le navigateur client pour les quatre dimensions de négociation de HTTP/1.1.

Dans la suite, nous nous intéressons à la méthode de négociation dite “dirigée par le serveur” qui est implémentée en Apache.

Les **entrées** de l'algorithme qui est à la base de cette méthode sont les suivantes :

– `V1-Vn` est la liste de variantes pour la ressource.
  Cette liste provient soit d'un fichier de variante (*type-map*) soit en utilisant les extensions des fichiers qui partagent un préfixe commun avec la ressource et se trouvent dans le même répertoire (voir la dernière partie du sujet pour un exemple).
  Chaque élément de la liste est un tuple ayant la structure suivante :

(URI, T, qsT, L, qsL, E, qsE, C, qsC)
Les champs T, L, E et C correspondent aux quatre dimensions de la négociation et peuvent être ''*' s'ils ne sont pas spécifiés. Les champs qsT, qsL, qsE et qsC correspondent aux facteurs de qualité "source" de chaque dimension de la variante. La valeur par défaut des facteurs de qualité est 1.

- Quatre listes, T1-Tn, L1-Ln, E1-En et C1-Cn qui correspondent respectivement aux préférences de types, de langues, d'encodage et d'ensemble de caractère indiquées par le navigateur.
  Ces listes sont construites à partir des informations (requête) envoyées par le navigateur client pour chaque dimension de négociation. Leur valeur par défaut est ((*, 1)).
  Chaque entrée de chaque liste est une paire ayant comme premier élément (nommé resp. T, L, E, C) un type MIME ou un modèle (*pattern*) de type MIME, et comme second élément (nommé q) le facteur de qualité associé (par défaut 1).

La **sortie** de l'algorithme est soit la variante qui correspond "au mieux" aux préférences exprimées, soit un message d'erreur.

L'**algorithme** appliqué par le serveur est le suivant :

```
variable LR : liste résultat de variantes
macro saut_3_4_cont(L) = si longueur(L) = 1 alors goto 3
                         si longueur(L) = 0 alors goto 4
début
LR <- V1-Vn
1: pour chaque V de V1-Vn faire
       pour chaque dimension D de { T, L, E, C } faire
           si V.D n'appartient pas à D1-Dn ou
              (V.D, 0.0) appartient à D1-Dn
           alors
                eliminer V de LR
           fin si
       fin pour
   fin pour
   saut_3_4_cont(LR)
2: // Selectionne la 'meilleure' variante de LR par élimination.
   2.1: Sélectionne dans LR les variantes pour lesquelles le produit
        T1-Tn(V.T).q * V.qsT est maximum.
        saut_3_4_cont(LR)
   2.2: Sélectionne dans LR les variantes avec L1-Ln(V.L).q maximum.
        saut_3_4_cont(LR)
   2.3: Sélectionne dans LR les variantes avec la meilleure correspondance
        de langue définie par l'ordre dans la liste L1-Ln ou dans la
        directive LanguagePriority.
        saut_3_4_cont(LR)
   2.4: Sélectionne dans LR les versions avec la version ('level') de type
        la plus haute (pour text/html).
        saut_3_4_cont(LR)
   2.5: Sélectionne dans LR les variantes avec le meilleur C1-Cn(V.C).q
        Par défaut, on suppose ISO-8859-1 avec q=1 si pas explicitement exclus.
```

```
    2.6: Sélectionne les variantes de LR qui n'ont pas comme ensemble de
         caractères ISO-8859-1. S'il n'existe pas de telles variantes,
         sélectionne toutes les variantes de LR.
    2.7: Sélectionne dans LR les variantes avec le meilleur E1-En(V.E).q
         S'il existe de telles variantes, ne sélectionne que celles-ci.
         S'il y a des variantes encodées (mais pas préférées) et non-encodées,
             ne sélectionne que les variantes non-encodées.
         Si toutes les variantes sont soit encodées soit non-encodées,
             sélectionne toutes les variantes.
         saut_3_4_cont(LR)
    2.8: Sélectionne les variantes avec le plus petit contenu.
         saut_3_4_cont(LR)
    2.9: Sélectionne dans LR la première variante en prenant comme ordre
         soit celui donné dans le fichier de variantes (s'il existe) soit
         l'ordre alphabétique.
3: // On arrive ici avec une liste LR de longueur 1
   Envoie first(LR) avec le code de succes.
   return
4: // On arrive ici si aucune variante n'est acceptable
   Envoie code d'erreur.
   return
fin
```

Questions :

1. Donner la liste `V1-Vn` construite à partir du fichier de variantes donné comme exemple au problème 3.
2. Appliquer l'algorithme pour la liste de variantes `V1-Vn` construite ci-dessus et les listes de dimensions de négociation suivantes :

   ```
   T1-Tn = ( (text/html,1), (image/gif,0.8), (text/plain,0.5), (*/*,0.01) )
   ```

   Indiquer la réponse du serveur en précisant les dimensions de la négociation prises en compte par l'algorithme.
3. La même question ci-dessus pour les listes de dimensions suivantes :

   ```
   T1-Tn = ( (text/*, 0.8), (image/*, 0.8), (application/*, 1) )
   L1-Ln = ( (en, 0.8), (fr, 1), (de, 1) )
   E1-En = ( (Z, 1), (*, 0.2) )
   ```

4. Expliquer (en français) quel est le but des pas `1` et `2.8` de l'algorithme.
5. Certains navigateurs célèbres ajoutent systématiquement (mais incorrectement par rapport à la norme) le modèle `*/*` à la fin de la liste des types acceptés. Par exemple, l'en-tête :

   ```
   Accept: text/html, text/plain, image/gif, */*
   ```

   est utilisé (incorrectement) pour indiquer que les types énumérés explicitement sont préférables, mais toute autre représentation disponible est acceptable.

Quel est le résultat de l'algorithme donné pour ce type de listes ?

En fait, l'algorithme donné est modifié en Apache pour palier à ce genre d'erreurs comme suit : un facteur de qualité de 0.01 est associé aux modèles `*/*` quand il apparaissent dans des listes de type sans facteurs de qualité. Quel effet a cette modification de l'algorithme de base ? Pourquoi cette modification est faite uniquement pour les listes sans facteur de qualité ?

6. Un problème opposé à celui ci-dessus apparaît pour la négociation des langues. En effet, si le navigateur spécifie comme préférences `en-GB ;g=0.9, fr ;q=0.8` et les versions disponibles sont `en` et `fr`, la version `fr` sera envoyée. Pour pallier à ce problème, l'algorithme de base est modifié pour ajouter à la liste des langues du navigateur la liste des langues "parent" (`en` pour `en-GB`) avec un très faible facteur de qualité (0.01).

   Quel effet a cette modification sur l'exemple donné ci-dessus ? Donner un autre exemple où cette modification s'applique.

7. Donner les lignes à insérer en `httpd.conf` pour que le propriétaire du répertoire `$www/pub/doc` puisse décider d'utiliser la négociation de contenu. Indiquer comment (fichier de configuration et directives) l'utilisateur peut ensuite choisir d'utiliser ce mécanisme avec un fichier de variantes.

3. HTML, CGI, PHP (5 points)

1. Ecrire un CGI (de préférence en PHP) qui reçoit comme unique argument l'URI d'une ressource et qui renvoie comme résultat le fichier des variantes de la ressource dans le format *type-map* (texte).

   Par exemple, si l'URI donné en argument est `http://www.serveur.com/pub/toto`, et dans le répertoire `$www/htdocs/pub` du serveur HTTP `www.serveur.com` la commande `ls toto.*` donne comme résultat :

   `toto.en.html   toto.gif.gz.de   toto.txt.fr`

   le fichier de variante généré doit être le suivant :

```
URI: toto

URI: toto.en.html
Content-language: en
Content-type: text/html

URI: toto.gif.gz.de
Content-type: image/gif
Content-enconding: gz
Content-language: de

URI: toto.txt.fr
Content-type: text/plain
Content-language: fr
```

   Supposer que vous disposez d'un tableau associatif qui indique pour chaque extension de fichier la dimension de négociation et le type MIME qui lui correspondent.

2. Ecrire la page HTML qui permet d'appeler le CGI programmé ci-dessus.