Sujets des exercices dirigés
Technologie pour les applications client-serveur
UE RSX 102
Année 2012-2013

Responsable :
A . Wei

Responsables historiques :
G. Florin et E. Gressier-Soudan

Précédents contributeurs à ce patrimoine pédagogique de G. Florin, S. Natkin and L. Duchien.

ED1 – Sockets

Exercice 1 : Utilisation du niveau transport TCP avec l'interface Socket

On étudie une application de commerce électronique, qui consiste à obtenir d'un site serveur distant une cotation pour une valeur boursière. La solution en mode message utilise le niveau transport TCP avec l'interface socket.

Ce problème ne traite que des comportements d'un client. Il n'est pas nécessaire de comprendre très en détail les codes présentés en C ou en C++ pour répondre aux questions.

L'application considérée est celle d'un mode client serveur basique avec un message requête de demande de cotation et un message réponse contenant le cours de bourse. On identifie donc comme premier élément du protocole (message et donnée à échanger dans le message) une requête qui comporte un nom de valeur boursière à coter. C'est une chaîne de caractères de longueur variable. La longueur est codée sur un entier long (valeur maximum 100 octets). La réponse comporte la cotation demandée (codée pour simplifier sous la forme d'un entier long). Elle comporte aussi un code réponse entier au cas où des erreurs auraient rendu l'opération impossible. Les structures de données échangées dans les messages en langage C sont donc décrites comme suit :

```
#define MAXSTOCKNAMELEN 100

struct Quote_Request
{
long len; /* Longueur de la requête */
char name[MAXSTOCKNAMELEN]; /* Nom de la valeur à coter */
};
struct Quote_Response
{
long value; /* Cours de la valeur */
long errno; /* 0 si succès, code d'erreur sinon */
};
```

On rassemble, dans une procédure utilisée par le client (connect_quote_server), les instructions nécessaires pour la mise en connexion du client avec le serveur, selon les primitives de l'interface socket.

- En entrée de la procédure, server est une chaîne de caractères qui définit le nom du service de cotation.

- En entrée de la procédure, port contient le numéro de port du service de cotation.

- En résultat HANDLE est un entier qui contient la référence du descriptif de la socket.

```
typedef int HANDLE;

HANDLE connect_quote_server (const char server[], u_short port)
{
struct sockaddr_in addr;
struct hostent *hp;
```

```
HANDLE sd;

/* Création de la terminaison locale */
sd = socket (AF_INET, SOCK_STREAM, 0);

/* Détermination adresse du serveur */
hp = gethostbyname(server);

/* Remise à zéro de la zone adresse et initialisation adresse du serveur */
memset ((void *) &addr, 0, sizeof addr);
addr.sin_family = AF_INET;
addr.sin_port = htons (port);
memcpy (&addr.sin_addr, hp->h_addr, hp->h_length);

/* Ouverture de la connexion avec le serveur */
connect (sd,(struct sockaddr *)&addr, sizeof addr);
return sd;
}
```

1) A quoi sert la primitive socket ? Pourquoi utiliser le paramètre AF-INET ? Pourquoi utiliser le paramètre SOCK-STREAM ?

2) A quoi sert la primitive gethostbyname. Quel est le nom de l'application Internet qui est utilisée par la primitive gesthostbyname. Quel doit donc être le format d'un nom de service ?

3) Quel est le rôle de la primitive connect ?

Une seconde procédure utilisée par le client send_request rassemble toutes les instructions nécessaires pour la transmission de la requête.

- En paramètre entrée de la procédure, sd est le descriptif de la socket.
- En paramètre entrée stock_name contient le nom de la valeur boursière à coter.
- Il n'y a pas de résultat (void).

```
void send_request (HANDLE sd ,const char stock_name[])
{
struct Quote_Request req;
size_t w_bytes;
size_t packet_len;
int n;

/* Détermination longueur du nom de valeur boursière et */
/* recopie du nom de valeur boursière dans la requête */
packet_len = strlen (stock_name);
if (packet_len > MAXSTOCKNAMELEN)
packet_len = MAXSTOCKNAMELEN;
strncpy (req.name, stock_name, packet_len);

/* Calcul longueur totale de la requête et conversion vers l'ordre des octets réseau */
packet_len = packet_len + sizeof( req);
```

```
req.len = htonl (packet_len);

/* Envoyer le message au serveur */
n = send (sd, ((const char *) &req),packet_len , 0);
}
```

4) Dans la procédure précédente send_request la primitive htonl est appliquée à la longueur du paquet packet_len pour fabriquer la zone req.len (longueur de la requête). Htonl (littéralement 'host to network long integer') convertit l'ordre des octets de la machine client dans l'ordre réseau. Pourquoi effectuer une telle conversion. Sous quel nom est connu le problème résolu par htonl?

On rassemble dans la procédure client recv_response la collecte d'un message de réponse à une requête de cotation. La structure de donnée d'un message de réponse Quote_Response a été définie plus haut.

```
int recv_response (HANDLE sd, long *value)
{
struct Quote_Response res;
recv (sd, (char*) &res, sizeof res, 0);
errno = ntohl (res.errno);
if (errno > 0) /* Erreur */
return -1;
else { /* Succès */
*value = ntohl (res.value);
return 0;
        }
}
```

5) A quoi servent les primitives ntohl utilisées dans le code recv_response ?

6) On souhaite évaluer la solution proposée de programmation client-serveur en mode message avec les sockets relativement à l'assemblage des données dans les messages. Quel est le point de vue adopté relativement aux conversions dans ce programme (quel est le niveau d'interopérabilité de la solution) ?

Exercice 2 :

Première partie :

Décrivez les deux procédures de l'Interface Socket via TCP et UDP. Donnez une comparaison entre ces deux procédures.

Deuxième partie :

On s'intéresse d'abord à la communication entre le serveur (www.cnam.fr) et le client (nom de la machine = Ulysse).

La structure de l'adressage Internet est la suivante :

```
struct sockaddr_in {
  uint8_t        sin_len;      /* longueur totale      */
  sa_family_t    sin_family;   /* famille : AF_INET    */
  in_port_t      sin_port;     /* le numéro de port    */
  struct in_addr  sin_addr;    /* l'adresse internet   */
  unsigned char   sin_zero[8];  /* un champ de 8 zéros  */
};
```

Si on utilise la primitive *gethostbyname*, on a la structure de désignation suivante :

```
    struct hostent {                  char *h_name;
                                      char **h_aliases;
                                      int h_addrtype;
                                      int h_length;
                                      char **h_addr_list; };
```

Du côté serveur, il doit réaliser les tâches suivantes : 1) ouvrir une socket ; 2) attendre des connexions et 3) fermer la socket.
Remplissez les parties manquantes dans le programme.

```
/* le serveur */

#include <stdio.h>
#include <unistd.h> // la function close()
#include <sys/types.h> // socket(), connect(), sendto(), recvfrom()
#include …….
#include ……
#include <string.h> // la function memset()
#include <stdlib.h>  // exit ()

#define MAXMESSAGE  1000

 int main (int argc, char *argv[]) {
  int Sock_Serv;
  struct sockaddr_in              Addr_Serv;
  int taille_Serv = sizeof (Addr_Serv);
  unsigned short Numero_port;

 char Message [MAXMESSAGE+1] ;

 //le numéro du port est saisi par le premier paramètre du programme. Atoi() convertir une
// chaine des caractères en un nombre entier
Numero_port = atoi (argv[1]);

// créer une socket UDP
  if ((Sock_Serv = socket(….,….,….)) < 0) {
    perror("Erreur d'ouverture du socket");
    exit (1);
  }
```

```
  // construire l'adresse locale.  Rappelons que le serveur attend des connexions venues d'un
//client. Dans ce cas,  le client pourrais avoir l'adresse «INADDR_ANY»
  Addr_Serv.sin_family      = ….;
  Addr_Serv.sin_addr.s_addr = …;
  Addr_Serv.sin_port        = htons( ..... );

  // construire l'adresse du serveur
//créer le primitive bind ()
  if (bind( ......., ..... , ..... ) < 0) {
    perror("erreur sur bind()");
    exit (1);
  }

//  attendre des messages de clients
While (1) {
// Mise à zéro du tampon « message » de façon à connaître le délimiteur de la chaîne.

memset(Message, 0, MAXMESSAGE);

if ((recvfrom(…..,…. ,…. ,…..,…. ,….,) < 0))
perror ("Problème de recvfrom ()");
exit(1) ;
}
printf ("message reçu : %s\n", Message);

close (Sock_Serv);
exit (0) ;
}
```

On s'intéresse ensuite au code du côté client. Le client doit réaliser les tâches suivantes : 1) créer une socket ; 2) envoyer un message (une requête) et 3) fermer la socket

Remplissez les parties manquantes dans le programme.

```
/*le client */

#include <stdio.h>
#include <unistd.h> //close ()
#include …..
#include …….
#include ……..
#include <string.h>
#include <stdlib.h> //exit(), atoi()


#define MAXMESSAGE  1000
#define PORT_SERVEUR 50 000

int main (int argc, char *argv[]) {
int Sock_Client;
```

```
struct sockaddr_in Addr_Serv;
struct hostent *hostclient = NULL ;
const char *hostname = "www.cnam.fr";

  char  EnvoieMessage [MAXMESSAGE+1] ;

  /* creation d'une socket */
  if ((Sock_Client = socket(…., .... , ....)) < 0) {
      perror("Erreur d'ouverture de la socket");
      exit (1);
  }

  // Construction de l'adresse qui comprend plusieurs champs :
    //  l'adresse IP du serveur à l'aide de la primitive « gethostbyname »
     //le numéro de port du serveur

hostclient = gethostbyname(hostname);
if (hostclient == NULL)
{
    printf  "problem de l'URL %s.\n", hostname);
     exit(1);
}

memcpy((char *) &Addr_Serv, 0, sizeof(Addr_Serv));
Addr_Serv . sin_family      = ;
Addr_ Serv.sin_addr.s_addr = ;
Addr_ Client.sin_port       = ;


// Envoyer un message */
  memset(EnvoieMessage, 0, MAXMESSAGE);

  memset(EnvoieMessage, " un message", sizeof(EnvoieMessage));

if (sendto(                ,          ,              ,      ) < 0)
  perror("erreur de sendto()");


 /* demander la fermeture*/
  …;
  exit (0);

}
```

ED2 - Appel de procédure distante

Exercice 1 : Parallélisme et performance des communications en RPC

Un appel de procédure distante présente les caractéristiques temporelles suivantes (valeurs données à titre d'exemple):

Traitements client de préparation d'appel: 5 ms

- Traitements RPC émission de la requête coté client
  Traitements de la souche client: 0,5 ms
  Traitements logiciels réseaux coté client: 0,3 ms
- Réseau
  Acheminement d'un message entre le client et le serveur 3 ms
- Traitements RPC réception de la requête coté serveur
  Traitements logiciels réseaux coté serveur: 0,3 ms.
  Traitements de la souche serveur: 0,5 ms

Traitements serveur de l'appel (exécution de la procédure): 10 ms

- Traitements RPC émission de la réponse coté serveur
  Traitements de la souche serveur: 0,5 ms
  Traitements logiciels réseaux coté serveur: 0,3 ms
- Réseau
  Acheminement d'un message entre le serveur et le client 3 ms
- Traitements RPC réception de la réponse coté client
  Traitements logiciels réseaux coté client: 0,3 ms.
  Traitements de la souche client: 0,5 ms

Un client souhaite exécuter sur un même serveur deux appels de procédures distantes indépendantes (non reliées) ayant les caractéristiques temporelles précédentes.

1) On fait l'hypothèse que les RPC sont synchrones. Rappelez la définition d'un RPC synchrone.

2) On fait l'hypothèse que client et serveur travaillent séquentiellement. Le client exécute les deux appels au moyen d'un seul processus séquentiel. La procédure distante est exécutée pour les deux fois par le même processus. Quelle est la durée totale nécessaire à la réalisation des deux appels?

3) On cherche maintenant une exécution parallèle efficace en utilisant le parallélisme chez le client et chez le serveur . La procédure distante est toujours en mode synchrone. Le client et le serveur utilisent des processus sur la machine client et la machine serveur ("multi processing"). Les appels sont donc réalisés sur le client au moyen de deux processus parallèles associés aux deux appels indépendants. Les traitements sont réalisés sur le serveur au moyen de deux processus lancés en parallèle associés à chaque appel de procédure distante. On néglige les temps de commutation de contexte entre les processus. On se place dans le cas de machines mono processeur c'est à dire que les exécutions lancées en parallèles sont exécutées par un seul processeur (pseudo parallélisme).

Dessinez le diagramme d'ordonnancement des opérations dans le cas de l'ordonnancement parallèle le plus efficace des opérations client et serveur. Représentez par des segments verticaux sur un dessin les différentes opérations selon les quatre fils d'exécution associés aux quatre processus Représentez également les messages de requête et de réponse? Quelle est la durée totale du traitement des deux appels?

4) On suppose maintenant que l'on dispose d'un RPC asynchrone. Rappelez la définition d'un RPC asynchrone ?

5) Le RPC asynchrone ne suppose pas l'existence de parallélisme sur le site client pour améliorer les performances. On considère dans cette question, que les clients et serveurs sont purement séquentiels. Le processus client réalise deux appels en RPC asynchrone et le processus serveur réalise les deux traitements des appels asynchrones. Dessinez l'ordonnancement le plus efficace des opérations pour les deux processus associés au client et au serveur. Représentez les messages échangés entre les deux processus. Quelle est la durée totale du traitement des deux appels ?

Exercice 2 : Traitement des pertes de messages

On souhaite réaliser très efficacement des appels de procédure distants en se plaçant non pas au dessus d'une couche transport fiable mais au dessus d'une couche réseau sans contrôle d'erreur.

Proposez des solutions efficaces au problème des pertes de messages d'appel et de réponse.

Exercice 3 : Exécutions très longues

On souhaite exécuter en mode appel de procédure distante des traitements de durée non prévisible (par exemple des interrogations distantes de bases de données dont les réponses peuvent prendre des durées très variables et très longues).

Proposez une solution pour le contrôle de l'exécution distante, en particulier pour pouvoir distinguer une panne de serveur d'une exécution longue.

Exercice 4 : Appels en parallèle (redondance massive)

L'appel distant active une seule procédure à distance à chaque fois. Un utilisateur peut souhaiter faire réaliser n exécutions à distance simultanément en lançant en diffusion une requête.

Explicitez le schéma du fonctionnement ainsi défini.

Utilisez le dans le cas de données répliquées. On suppose que l'on a n serveurs de données en copies multiples (pour la sécurité) qui peuvent réaliser les mêmes opérations lire et écrire. Comment sont réalisées les lectures, les écritures. Quelle propriété minimum doivent satisfaire les requêtes pour maintenir la cohérence des données si l'on considére qu'il n'y a pas de partage de données entre plusieurs activités?

Exercice 5 : Serveurs multiples (redondance sélective)

Dans certains types de service on souhaite disposer de plusieurs instances du même service sur plusieurs machines différentes (par exemple pour un service de compilation). L'appel distant active une seule exécution à distante à chaque fois.

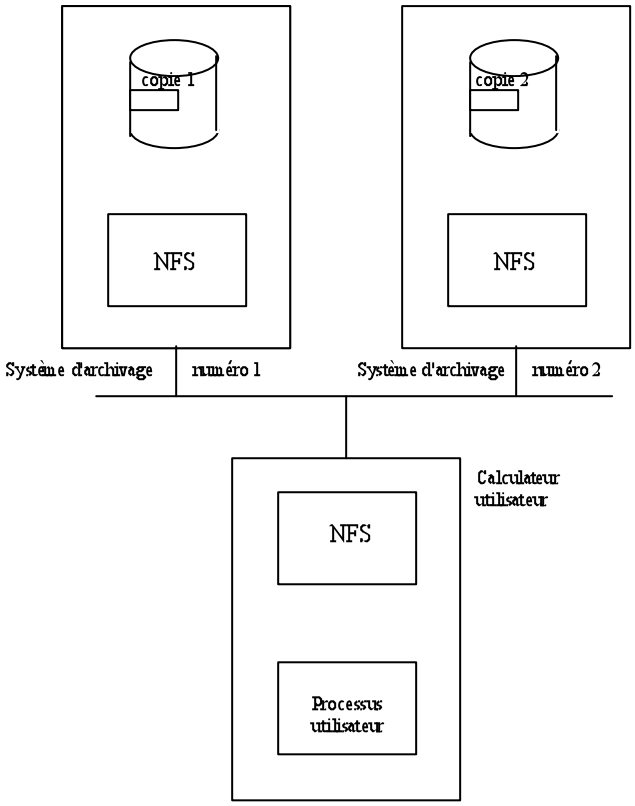
Dans quel but utiliserait-on de tels services à instances multiples?

Proposez une organisation pour rendre efficacement un tel service.

ED3 – NFS

Exercice 1 : Système de fichiers répartis NFS et réplication

On souhaite modifier le système de fichiers répartis NFS de manière à permettre pour certains fichiers d'un type particulier l'existence de plusieurs copies des disques différents afin d'avoir une meilleure tolérance aux pannes.
On aura donc par exemple la configuration suivante pour deux copies:



Dans tout le texte qui suit on considère pour simplifier que les fichiers ne sont pas partagés entre plusieurs utilisateurs.

Question 1

1.1 Rappelez les principes généraux du fonctionnement de NFS

1.2 NFS est un système de fichiers répartis "sans état". Rappelez les principes essentiels du fonctionnement "sans état" en particulier comment sont exécutées les requêtes successives sur les systèmes d'archivages.

1.3 Le mode de fonctionnement "sans état" autorise une technique simple de traitement des pannes du serveur au niveau du RPC. Quelle est -elle ? Expliquez pourquoi elle est tout à fait adaptée au mode "sans état".

Question 2

Dans le système de fichiers répliqués basé NFS proposez une méthode efficace pour la réalisation des opérations de lecture.

Question 3

Proposez une méthode de réalisation de l'opération d'écriture qui utilise une diffusion de la requête à tous les systèmes d'archivages.

Exercice 2 :Désignation des fichiers dans le système de fichiers répartis NFS

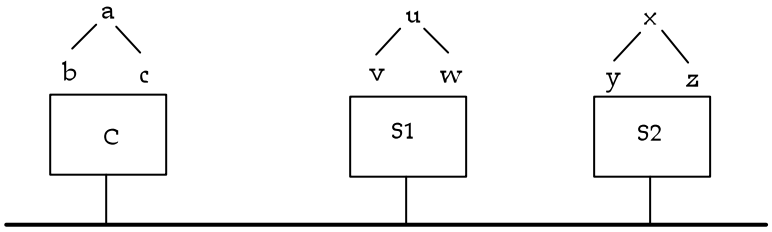
NFS (Network File system) est un système de gestion de fichiers répartis. L'arborescence de fichiers, répartie physiquement sur plusieurs machine et accessible par l'usager depuis son poste de travail, est construite en utilisant une technique de montage d'arborescence. La vision qu'a un utilisateur de l'arbre des fichiers est donc une vue logique et locale à sa machine de travail.

1 - Rappelez brièvement les principes du montage d'arborescence .

2 - Les principes de désignation des fichiers dans NFS découlent de l'utilisation du montage d'arborescence. Rappelez brièvement ces principes de désignation sur lequel s'appuie NFS .

3 - Quels sont les avantages et les inconvénients de ce système de désignation ? .

Soit la configuration :



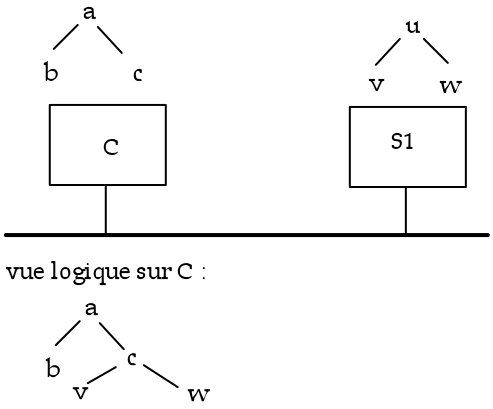
On dispose des opérations suivantes :

. une opération "monter" permet d'attacher des arborescences d'une machine serveur sur une machine client:

monter <serveur> : <chemin absolu d'un répertoire> sur <chemin absolu d'un répertoire local au client>

exemple : sur le site C, l'opération monter S1:/u sur /a/c

donne :



Lors du montage, les deux nœuds u et c coïncident. C'est le nœud local (ici c) qui impose son nom, mais le contenu de c correspond maintenant au contenu de u (v et w ici), les anciens fichiers de c ne sont plus visibles.

. une opération qui permet de lister le contenu d'un répertoire :

contenu <chemin d'accès à un répertoire>

Rappelons que comme dans tout système de fichiers arborescent un chemin d'accès peut être défini en absolu. Par exemple sur la machine C si le répertoire courant est b, **"contenu /a/c"** donne à l'utilisateur ce que contient le répertoire u soit **"v,w"**. Le chemin d'accès peut être défini en relatif par rapport à la position courante. La position courante étant le répertoire a, la commande **"contenu ./c"** donne le **même résultat** que précédemment (le point "." désigne le répertoire courant d'où l'utilisateur exécute la commande).

4 - On considère la suite de commandes 1

Sur le site C : monter S1:/u sur /a/c

Sur le site C : monter S2:/x sur /a/c/w

Quel est le résultat de cette suite de commandes ? Quelle est l'arborescence vue par le client C? Commentez votre réponse. Quel est le résultat de la commande "contenu /a/c "?

5 - On considère la suite de commandes 2 (par rapport à la même situation initiale que la question précédente)

Sur le site S1 : monter S2:/x sur /u/w

Sur le site C : monter S1:/u sur /a/c

Quel est le résultat de cette suite de commandes vu par le client C. Commentez votre réponse. Quel est le résultat de la commande "contenu /a/c" ?

6 - On cherche les principes d'un algorithme permettant le parcours du chemin d'accès à un fichier (celui-ci étant défini en absolu)

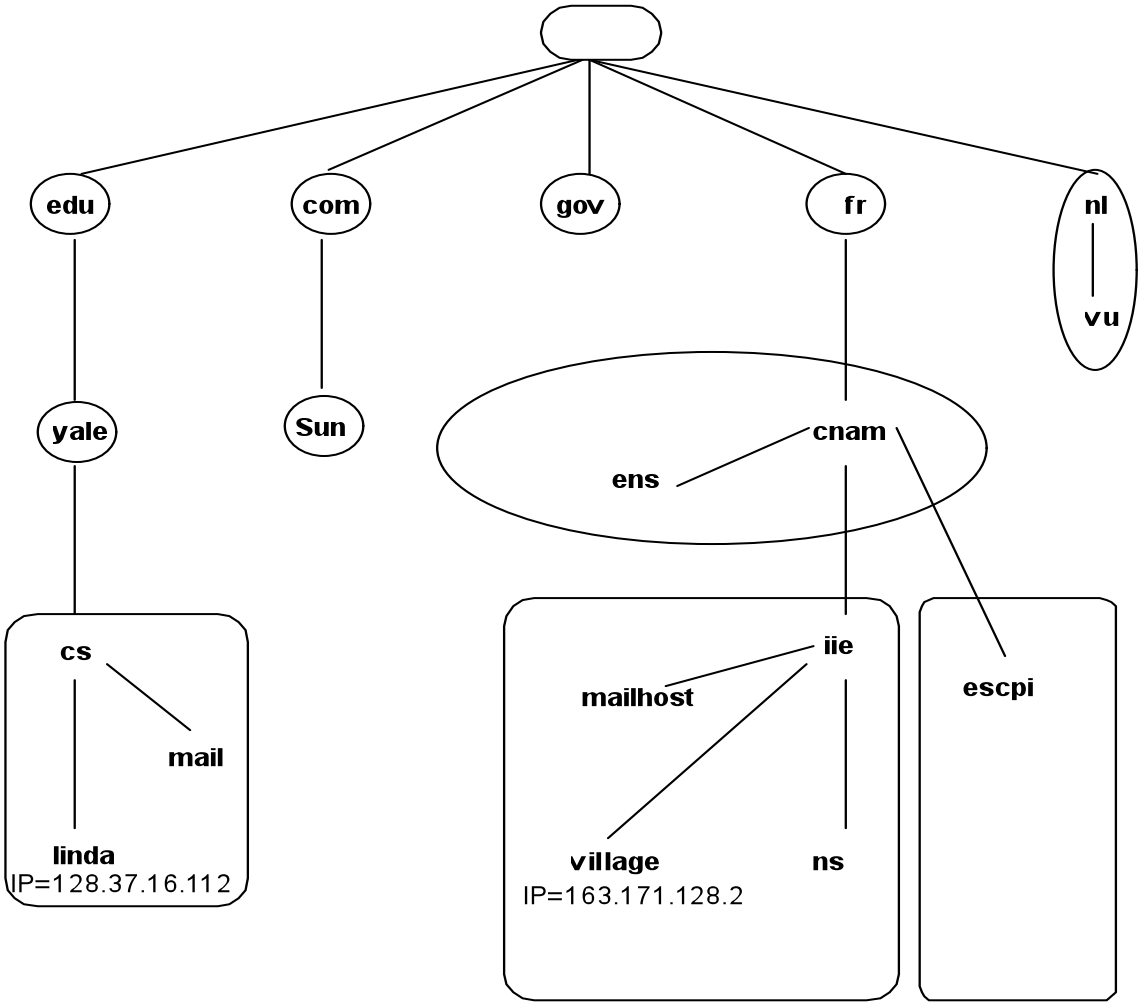
6. 1 - Donnez brièvement les principes d'une solution itérative de parcours d'un chemin d'accès à un fichier.

6. 2 - Donnez brièvement les principes d'une résolution récursive du chemin d'accès à un fichier. Sachant que NFS utilise une technique de montage d'arborescence, et qu'une machine client NFS conserve en mémoire la liste des points de montage, que pensez-vous de l'application d'une méthode de résolution récursive dans NFS.

ED4 – DNS

Exercice 1 : DNS "Domain Name System"

Un sous ensemble de la hiérarchie de l'espace de nommage Internet est donné par la figure suivante dans laquelle les ensembles encadrés définissent des zones d'administration. On trouve par exemple sur ce dessin une machine dont l'adresse IP est 163.171.128.2 et dont le nom est village.iie.cnam.fr .



L'une des utilisations importantes du DNS est la détermination de l'adresse IP d'un site Internet permettant d'envoyer du courrier électronique à l'un de ses usagers. Une personne possède l'adresse de courrier électronique suivante: asterix@village.iie.cnam.fr.

1 Rappelez le nom du protocole Internet qui réalise la messagerie électronique?

2 Rappelez le rôle du serveur de courrier dans un domaine Internet. Donnez le nom DNS de la machine qui est le serveur de courrier électronique de l'usager asterix (le nom le plus probable sur le dessin)?

3 Lorsque le client de messagerie de l'utilisateur superman@linda.cs.yale.edu envoie un courrier au serveur de messagerie de l'utilisateur asterix@village.iie.cnam.fr, que fait-il tout d'abord?

4 Rappelez le rôle du serveur de noms dans un domaine Internet. Donnez le nom de la machine serveur de noms qui administre le sous domaine de la machine village (le nom le plus probable)?

5 La détermination des adresses IP à partir des noms logiques est réalisée sur requête d'un site demandeur (le résolveur) à son serveur de noms. Les serveurs dialoguent selon le protocole DNS. On rappelle qu'un serveur de noms DNS possède une base de données dont les enregistrements sont de la forme:

Nom_de_domaine, Durée_de_vie, Type, Classe, Valeur

- Nom_de_domaine est un nom DNS
- Durée_de_vie détermine en secondes, la durée de validité de l'information.
- Classe indique le protocole concerné. Elle est ici toujours égale à IN pour Internet.
- Valeur est une donnée caractéristique du type et du nom de domaine.
- Type définit le type de l'enregistrement stocké ('resource record'):

  A: la valeur est alors l'adresse IP de l'hôte désigné dans le champ Nom_de_Domaine

  MX: la valeur est alors la priorité (entier dans 1..n) et le nom du serveur de courrier pour l'hôte désigné dans le champ Nom_du_domaine (un hôte peut avoir plusieurs serveurs de courrier).

  NS: la valeur est alors le nom de domaine d'un serveur de noms DNS pour le domaine considéré.

Donnez deux enregistrements de la base de données DNS du domaine iie sachant que leur durée de vie est un jour ?

6 En pratique, les serveurs DNS envoient des messages qui contiennent plusieurs requêtes ou plusieurs réponses.

Quel est le nom de cette technique? Quel est son utilité ?

7 Lorsque qu'un serveur DNS apprend une information suite à une requête utilisateur, il enregistre cette information en mémoire temporaire si elle ne concerne pas son domaine.

7.1 Quel est le nom de cette technique ?
7.2 Citez deux avantages importants de cette technique dans le cas du DNS ?
7.3 Pourquoi avoir introduit un champs Durée_de_vie associé à chaque information DNS ?

8 Le client de messagerie de l'utilisateur superman@linda.cs.yale.edu utilise la résolution itérative du DNS pour atteindre l'utilisateur asterix@village.iie.cnam.fr. Cette résolution itérative d'une requête concernant un nom de domaine par le DNS implique un ensemble de serveurs DNS. Rappelez la stratégie de résolution utilisée. Donnez pour l'exemple précédent les noms des domaines concernés et le schéma de parcours.

9 On améliore l'apprentissage des résolutions de requêtes par une seconde technique de résolution baptisée résolution récursive. Contrairement à la résolution itérative la réponse fait le chemin inverse de la requête et est donc apprise par tous les sites

traversés. Dans le cadre de cette dernière solution, lorsque superman@ linda.cs.yale.edu envoie un courrier à asterix@village.iie.cnam.fr, quelle est la liste des serveurs consultés et quel est le parcours des messages ?

Exercice 2 : DNS "Domain Name System"

1 Dans le réseau Internet le système de gestion des noms de domaines DNS (‘ Domain Name System’) joue un rôle très important. Quel est l’objectif général du DNS? Citez quatre services différents rendus par le DNS aux applications Internet.

2 Qu’est ce qu’un domaine DNS? A quel domaine appartient le nom pollux.escpi.cnam.fr?

3 Qu’est ce qu’une zone DNS? A quelle zone appartient probablement le nom pollux.escpi.cnam.fr?

4 Citez quelques règles qui président à la structuration des noms (à la façon dont les noms sont formés).

5 Les noms de domaines sont créés pour des besoins différents. Citez quelques règles qui président au choix des noms par les usagers (les usagers peuvent-ils choisir n’importe quel nom pour un domaine ou pour un service)?

6 Dans des bases de données DNS on trouve des définitions d’enregistrements associés aux noms suivants : www.discount.com
6.17.173.163.in-addr.arpa
ulysse.cnam.fr
Quel est le motif le plus probable qui conduit à la création de ces noms de domaine?

7 Compte tenu de la signification proposée à la question précédente citez le type d’un enregistrement (RR ‘Resource Records’) d’une base de données DNS associée à chacun de ces noms.

8 La résolution d’une requête adressée au DNS par une application Internet s’effectue via un résolveur (c’est à dire une procédure prédéfinie comme ‘gethostbyaddr’ en UNIX appelée dans un programme) et un ensemble de serveurs de noms. Quand on utilise une approche récursive de résolution de requête quel est le statut du résolveur et quel est le statut d’un serveur. Sont ils client, serveur ou les deux à la fois?

9 Quel est le statut du résolveur et quel est le statut d’un serveur (client, serveur ou les deux à la fois) dans une résolution itérative?

10 Un utilisateur d’un navigateur WEB définit l’URL suivante : ‘http://www‘. Quelle est la première opération effectuée par le navigateur sur la chaîne www? Que se passe t’il dans le cas présent relativement à cette chaîne?

11 Quel mécanisme proposez vous pour garantir qu'une information obtenue par le DNS a bien été générée par l'administrateur du domaine?

12 Comment intégrez vous au DNS un mécanisme de distribution de clés publiques?

Exercice 3 : Utilisation d'un résolveur DNS

Pour mieux comprendre les différents noms et adresses utilisés dans un courrier, un résolveur interactif DNS a été lancé depuis un poste de travail. L'outil utilisé ici est nslookup qui est activé sous Microsoft Windows 2000 dans la fenêtre invite de commande par l'échange suivant. Le résultat est le copié collé de la fenêtre, fautes d'orthographes dans nslookup francisé Windows comprises.

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>nslookup
Serveur par défaut : dns2.proxad.net
Address:  212.27.54.252
```

La première recherche effectuée concerne le nom de domaine topachat.com selon le type A. Nous allons voir dans les exemples que nos recherches dans le DNS effectuées avec nsloolup sont toujours précédées d'une directive qui fixe le type des enregistrements ressources recherchés (les RR ou 'Resources Records'). La commande set q=xxxx est une abréviation pour set querytype=xxxx.

```
> set q=a
> topachat.com
Serveur :  dns2.proxad.net
Address:  212.27.54.252

R'ponse ne faisant pas autorit'ÿ:
Nom :    topachat.com
Addresses:  213.41.42.206, 213.41.42.210, 195.167.228.246, 213.41.42.216
213.41.42.202, 213.41.42.207, 213.41.42.203, 195.167.228.245, 195.167.
228.247
```

1) Commentez cette recherche

a) A quoi correspondent les lignes Serveur et Address ?

b) Pourquoi cette réponse a comme statut : 'Réponse ne faisant pas autorité' ?

c) Pourquoi le nom de domaine topachat.com est-il associé à ces nombreuses adresses IP (dans quel but) ?

La seconde recherche effectuée concerne le nom de domaine topachat.com selon le type MX. On avait aussi vu apparaître dans le courrier du problème précédent l'hôte mta1.topachat-clust.com qui jouait un rôle dans le courrier de cette entreprise.

```
> set q=mx
> topachat.com
Serveur :  dns1.proxad.net
Address:  212.27.53.252

R'ponse ne faisant pas autorit'ÿ:
topachat.com    MX preference = 10, mail exchanger = ns2.topachat-clust.com
topachat.com    MX preference = 20, mail exchanger = ns4.topachat.com
topachat.com    MX preference = 40, mail exchanger = ns0.topachat-clust.com

ns4.topachat.com        internet address = 195.167.228.241
ns0.topachat-clust.com  internet address = 213.41.42.199
ns2.topachat-clust.com  internet address = 213.41.42.198

> set q=a
> mta1.topachat-clust.com
```

```
Serveur :  dns2.proxad.net
Address:  212.27.54.252

R'ponse ne faisant pas autorit'ÿ:
Nom :    mta1.topachat-clust.com
Addresses:  213.41.42.197, 195.167.228.242
```

2) Commentez ces recherches

a) Pourquoi selon le type MX, le nom de domaine topachat.com correspond à un ensemble de noms de domaines ?

b) Le nom de domaine mta1.topachat-clust.com ne figure pas dans la liste de résultat pour le domaine topachat.com avec le type MX. Comment expliquer cela ?

La troisième recherche concerne le domaine cnam.fr.

```
> set q=mx
> cnam.fr
Serveur :  dns1.proxad.net
Address:  212.27.53.252

R'ponse ne faisant pas autorit'ÿ:
cnam.fr MX preference = 10, mail exchanger = brangien.cnam.fr

brangien.cnam.fr        internet address = 163.173.128.20
> set q=cname
> imap.cnam.fr
Serveur :  dns1.proxad.net
Address:  212.27.53.252

R'ponse ne faisant pas autorit'ÿ:
imap.cnam.fr    canonical name = copernic.cnam.fr
> set q=a
> copernic.cnam.fr
Serveur :  dns2.proxad.net
Address:  212.27.54.252

R'ponse ne faisant pas autorit'ÿ:
Nom :    copernic.cnam.fr
Address:  163.173.128.12
```

3) Commentez ces recherches effectuées pour confirmer des hypothèses relatives au courrier électronique du CNAM déduites du courrier utilisé au problème précédent.

a) Rappelez la définition d'un RR (enregistrement ressource) de type cname ?

b) On peut confirmer les informations relatives au rôle de brangien.cnam.fr et de copernic.cnam.fr vues dans le courrier électronique. Lesquelles ?

Une quatrième recherche concerne un courrier non sollicité dans lequel est apparu comme adresse source du courrier 66.63.190.176. Cette adresse est mentionnée comme inconnue dans ce courrier concernant une offre de cartouches d'encre (courrier non cité dans le sujet parce que trop volumineux). La recherche qui a été effectuée est la suivante :

```
> set q=ptr
> 176.190.63.66.in-addr.arpa..
Serveur :  dns1.proxad.net
Address:  212.27.53.252
```

```
R'ponse ne faisant pas autorit'ÿ:
176.190.63.66.in-addr.arpa      name = 66.63.190.176.oc3networks.com
> set q=any
> oc3networks.com..
Serveur :  dns1.proxad.net
Address:  212.27.53.252

R'ponse ne faisant pas autorit'ÿ:
oc3networks.com internet address = 66.63.160.51
oc3networks.com nameserver = ns2.oc3networks.com
oc3networks.com nameserver = ns1.oc3networks.com
oc3networks.com
        primary name server = ns1.oc3networks.com
        responsible mail addr = hostmaster.oc3networks.com
        serial  = 9
        refresh = 86400 (1 day)
        retry   = 3600 (1 hour)
        expire  = 432000 (5 days)
        default TTL = 3600 (1 hour)
oc3networks.com MX preference = 10, mail exchanger = mail.oc3networks.com

mail.oc3networks.com    internet address = 66.63.164.163
ns2.oc3networks.com     internet address = 66.63.164.173
ns1.oc3networks.com     internet address = 66.63.160.6
>
```

4) Commentez ces recherches.

a) Pourquoi fait-on une recherche de type ptr sur un nom de domaine de la forme 176.190.63.66.in-addr.arpa..?

b) Pourquoi l'adresse IP 66.63.190.176 avait elle été mentionnée dans le courrier électronique comme inconnue ('unknown') ?

c) L'information obtenue sur la source du courrier est qu'il provient d'un domaine dont le nom est oc3networks.com. Quels RR obtient-on en fait par la recherche sur le type any sur l'information obtenue sur la source du courrier oc3networks.com ?

ED5 - SMTP

Messagerie Internet SMTP

SMTP ('Simple Mail Transfer Protocol') est le nom d'ensemble donné à la messagerie normalisée par l'IETF ('Internet Engineering Task Force') pour le réseau Internet.

I Architecture de la messagerie SMTP

I.1) Une messagerie électronique (comme SMTP) est une application répartie. Définissez très brièvement les objectifs et les fonctions réalisées par une messagerie.

I.2) Une messagerie électronique et un transfert de fichiers (comme FTP 'File Transfer Protocol') définissent des services applicatifs voisins. Rappelez très brièvement les fonctions réalisées par un transfert de fichiers. Citez plusieurs similitudes et plusieurs différences entre une messagerie et un transfert de fichiers.

I.3) Une messagerie électronique et un service de files de messages (MOM 'Message Oriented Middlleware' comme le produit IBM MQ Series 'Message Queue Series') définissent également des services applicatifs assez voisins. Rappelez très brièvement les fonctions réalisées par des files de messages. Citez plusieurs similitudes et plusieurs différences entre une messagerie et un service de files de messages.

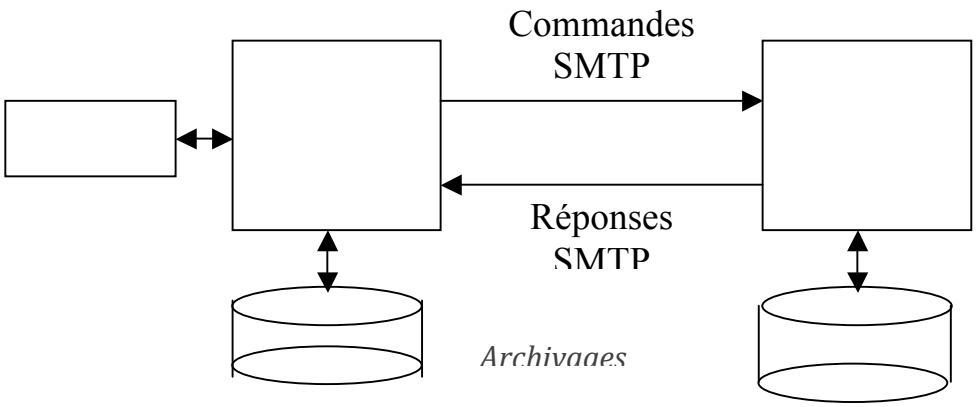
I.4) Il existe deux grandes catégories d'architectures pour les messageries. La première catégorie est basée sur un principe de délivrance de bout en bout ('end to end delivery'). La seconde catégorie est basée sur une approche de stockage et retransmission ('store and forward delivery'). Rappelez les principes de ces deux modes d'acheminement. Pour chaque mode, donnez un schéma avec les différents types de logiciels concernés.

I.5) Dans quelle catégorie se place la messagerie SMTP? Pour répondre vous devez absolument justifier votre réponse, en présentant des éléments d'architectures et de protocoles de la messagerie Internet?

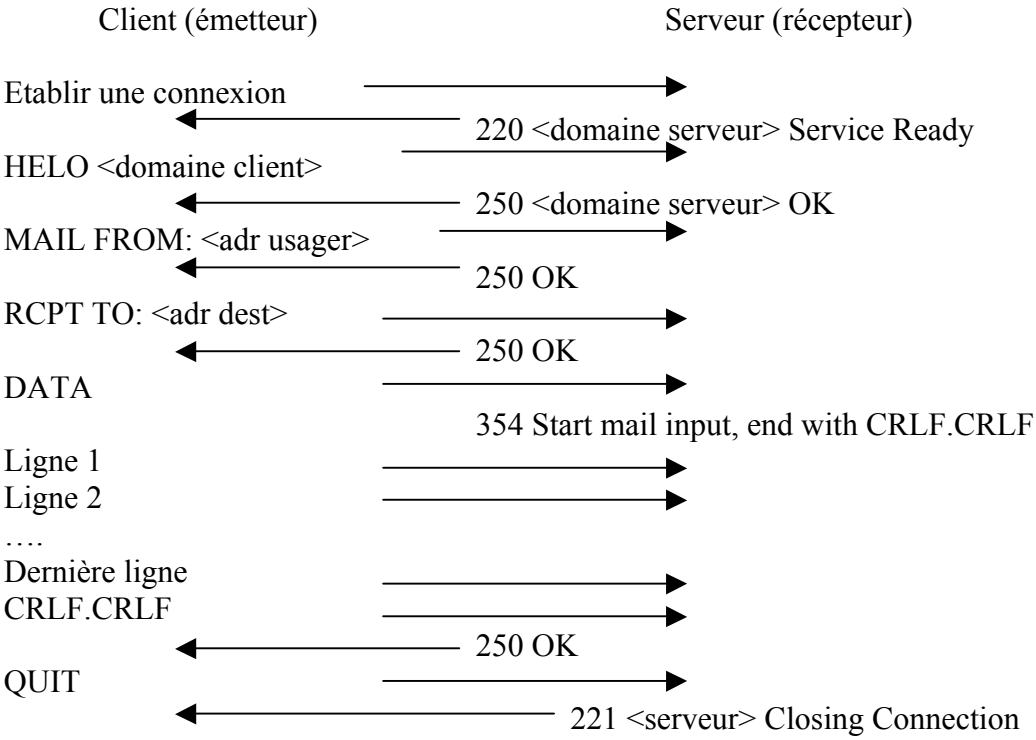
I.6) Quelles sont les différentes étapes de la transmission d'un courrier électronique dans le cas habituel d'un prestataire de service d'accès Internet ou d'une entreprise de taille significative. Précisez les matériels (ordinateurs), les logiciels (agent utilisateur, serveur de messagerie, donnez des exemples de ces logiciels) et les protocoles utilisés?

II Le protocole d'émission de courrier SMTP

Le protocole SMTP est un protocole client/serveur entre un client de messagerie (un émetteur de courrier) et un serveur de messagerie:



Le diagramme en flèches d'échange de messages suivant présente un cas de transmission d'un courrier d'un client vers un serveur de messagerie.



Les messages du client vers le serveur comportent un type. HELO est une ouverture de session entre le client et le serveur (le message contient le nom de domaine du client). MAIL FROM: définit l'adresse mail de l'usager émetteur pour le retour éventuel de diagnostics d'erreur. RCPT TO: définit l'adresse mail du destinataire, DATA définit l'enveloppe (l'entête) et le corps (le texte) du message. QUIT termine un courrier.

Pour chaque message on a un code de réponse sous la forme habituelle dans l'Internet c'est à dire avec un code numérique sur trois chiffres décimaux et une explication complémentaire en clair.

Un utilisateur souhaite faire fonctionner le protocole SMTP en générant par lui même les messages en direction d'un serveur de courrier connu et non verrouillé. Concrètement il créé un client de courrier (sous UNIX ou dans une fenêtre d'exécution de commandes Windows) en tapant la commande :

telnet cnam.fr 25

II.1) A quoi sert normalement l'utilitaire telnet?
II.2) A quoi correspond la chaîne cnam.fr. Quelle information doit être associée à cette chaîne par un ingénieur système pour que l'accès à un serveur de messagerie soit possible?
II.3) A quoi correspond le nombre 25?
II.4) Quels sont les différents aspects des protocoles telnet et smtp qui permettent de faire manuellement de la messagerie avec telnet?
II.5) L'utilisateur michel dans le domaine cnam.fr veut envoyer un courrier à l'utilisateur pierre dans le même domaine. Tout se passe bien. Donnez la liste des messages émis et des réponses reçues (chaque message a une réponse). C'est également la liste des lignes de textes qui s'affichent sur le terminal d'un essai manuel avec telnet (lignes de textes acheminées par

telnet dans le sens client vers le serveur et lignes de texte des réponses reçues du serveur vers le client).

II.6) Le protocole SMTP permet par ses différentes commandes à un serveur de messagerie de relayer du courrier. Expliquez pourquoi?

III Protocoles de réception de courrier

III.1) Un protocole de réception de courrier fonctionne entre un agent utilisateur et un serveur de messagerie et permet à l'utilisateur de lire son courrier. On distingue généralement trois modes de lecture du courrier. Rappelez les trois modes possibles ?

III.2) Les deux modes les plus utilisés sont le mode hors ligne associé au protocole POP et le mode en ligne associé au protocole IMAP. Citez pour chacun des deux protocoles des avantages et des inconvénients. En déduire les domaines d'application des deux protocoles?

III.3) Dans un outil implantant un client POP on trouve les paramètres suivants. A quoi servent ces paramètres?

SMTP server smtp.cnam.fr
POP server pop.cnam.fr
Mail account: natkin
Mail adress: natkin@cnam.fr
Reply adress: natkin@cnam.fr

Le protocole POP fonctionne en mode connecté avec TCP. Après avoir fourni un message d'acceuil après la connexion, chaque connexion POP passe par trois phases successives .

La phase d'authentification ('Authorization' en anglais)
La phase de transaction ('Transaction' en anglais)
La phase mise à jour ('Update' en anglais). Cette phase consiste à modifier définitivement l'archive de courrier gérée par le serveur en fonction des directives données par l'usager pendant le session (en particulier c'est à ce moment que sont détruites les copies indésirables).

Le client et le serveur POP dialoguent en utilisant des messages de commandes et de réponses. La syntaxe des principales commandes et réponses est donnée par la BNF suivante:

```
client_commands::="USER" mailbox CRLF/
                  "PASS" string CRLF/
                  "STAT" string CRLF/
                  "RETR" msg CRLF/
                  "DELE" msg CRLF/
                  "QUIT" CRLF

server_response::= simpleresponse /
```

statresponse/
errorresponse

simpleresponse::= "+0K" [" "*text] CRLF
[multiline CRLF "." CRLF]

statresponse::= "+0K" [" "1*DIGIT" "1*DIGIT]

errorresponse ::="-ERR"[" "*text]

multiline::=1*<n'importe quelle chaîne caractère ASCII ne comprenant pas la chaîne CRLF"."CRLF>
maibox::=string
string::=1*<n'importe quelle chaîne caractère ASCII ne comprenant pas les caractères SP, TAB et CRLF>
msg::=1*DIGIT
text::=1*<n'importe quelle chaîne caractère ASCII ne comprenant pas le caractère CRLF >

On retrouve ces trois phases dans l'exemple suivant du fonctionnement de POP

N°	sens	PDU échangée
1	S->C	+OK pop.cnam.fr server ready
2	C->S	USER natkin
3	S->C	+OK password required for natkin
4	C->S	PASS monpacs
5	S->C	-ERR password supplied for natkin is incorrect
6	C->S	USER natkin
7	S->C	+OK password required for natkin
8	C->S	PASS monpass
9	S->C	+OK maildrop has 1 messages (600 octets)
10	C->S	STAT
11	S->C	+OK 1 600
12	C->S	RETR 1
13	S->C	+OK
14	S->C	<contenu du message>
15	C->S	DELE 1
16	S->C	+OK
17	C->S	QUIT
18	S->C	+OK pop.cnam.fr signing off

Le contenu du message est le suivant:

X-Mailer: Microsoft Outlook Express Macintosh Edition - 4.5 (0410)
Date: Sat, 02 Jun 2001 15:26:38 +0200
Subject: Publi
From: "gerard" <gerard@cnam.fr >
To: natkin@cnam.fr
Mime-version: 1.0
X-Priority: 3

Content-type: multipart/mixed;
  boundary="MS_Mac_OE_3074340399_15662732_MIME_Part"

--MS_Mac_OE_3074340399_15662732_MIME_Part
Content-type: text/plain; charset="ISO-8859-1"
Content-transfer-encoding: quoted-printable

Ci joint les publications.

--
G=E9rard Florin
--MS_Mac_OE_3074340399_15662732_MIME_Part
Content-type: application/msword; name="publi.doc";
 x-mac-creator="4D535744";
 x-mac-type="5738424E"
Content-disposition: attachment
Content-transfer-encoding: base64

0M8R4KGxGuEAAAAAAAAAAAAAAAAAAAAAPgADAP7/CQAGAAAAAAAAAAA
AAAABAAAARQAAAA
AAAAEAAARwAAAAEAAAD+////AAAAAEQAAAD/////////////////////////////////

--MS_Mac_OE_3074340399_15662732_MIME_Part--

III.4) Donner pour chaque message échangé (chaque PDU échangée) dans l'exemple

d'échange précédent la phase à laquelle il appartient?

III.5) Pourquoi y a t'il une syntaxe spécifique pour la réponse à la commande RETR?

III.6) Dans l'exemple précédent dans quelle PDU et à quel usage est utilisé la définition de multiline?

III.7) Pourquoi la chaîne CRLF"."CRLF est-elle interdite dans la définition de multiline?

III.8) Expliquez ce que signifient

Content-type: multipart/mixed;
Content-type: application/msword; name="publi.doc";
Content-disposition: attachment
Content-transfer-encoding: base64

III.9) On souhaite définir l'automate de protocole d'un client POP capable de traiter la séquence de réception de courrier donnée précédemment en exemple. On identifie les cinq états suivants:

1. En attente
2. Préparation des données et ouverture de la connexion TCP vers le serveur
3. Authentification
4. Transaction: réception des messages et mise en forme
5. Mise à jour du serveur et fin des transferts.

Représentez l'automate possédant les cinq états.

III.10) Sur une machine donnée on dispose des appels de procédure suivants qui sont fournis par différents logiciels réseau.

gethostbyname(in:nom_de_machine, out:IP, out, statut)
listen()
connect()
send()
receive()
close()
convertbase64asciitobin(in:multiline, out : nom_de_fichier,out:statut)

Rappelez quels sont les différents logiciels qui fournissent les services précédents ?

III.11) Expliquez simplement ce qui est réalisé dans chaque état de l'automate (procédures de la liste précédente et commandes POP utilisées).

ED6 - Services et protocoles d'annuaires répartis LDAP

DAP ("Directory Access Protocol") est un protocole de gestion répartie de l'annuaire X500. LDAP ("LightWeight Directory Access Protocol") est un protocole simplifié dérivé de DAP permettant un accès à un annuaire en mode client/serveur en utilisant TCP/IP.

1) DNS "Domain Name System" est le service d'annuaire (ou de gestion de noms) de l'Internet. Rappelez quels sont les objectifs du DNS.

2) LDAP propose un protocole complet permettant d'accéder de façon transparente à un annuaire réparti partiellement dupliqué. Citez trois problèmes que doit résoudre un tel protocole en précisant pour chacun d'eux clairement sa nature.

LDAP est un protocole de niveau application de l'Internet (utilisant les services de TCP/IP). Il s'agit d'une version simplifiée de DAP permettant à un client LDAP d'accéder, de modifier, d'ajouter des données à un annuaire géré par un serveur LDAP. LDAP est décrit en ASN1.

On utilise les types suivants:

LDAPString ::= OCTET STRING

LDAPDN ::= LDAPString

LDAPDN (LDAP "Distinguished Name") définit des chaînes de caractères permettant de désigner une personne physique ou morale au moyen de différents attributs selon une syntaxe décrite dans la RFC 4511 (un exemple simple est donné plus loin).

La PDU bindRequest (PDU signifie une unité de données transportée par un protocole) permet d'ouvrir un dialogue entre le client et le serveur. Son contenu est défini par la spécification ASN1 suivante:

```
BindRequest ::=
        [APPLICATION 0] SEQUENCE  {
                version                 INTEGER (1 .. 127),
                name                    LDAPDN,
                authentication          CHOICE              {
                        simple          [0] OCTET STRING,
                        krbv42LDAP [1] OCTET STRING,
                        krbv42DSA               [2] OCTET STRING
                                                            }
                                                }
```

Le mode d'authentification 0 ne donne pas lieu à authentification si la chaîne est vide. Une authentification par mot de passe est utilisée si la chaîne n'est pas vide. Les deux autres modes sont liés à l'utilisation de SASL - Kerberos ou de DSA ("Digital Signature Algorithm").

3) Que signifient dans la description précédente les expressions APPLICATION 0, SEQUENCE, INTEGER, CHOICE?

La spécification ASN1 suivante décrit l'ensemble des messages (PDU) échangés par le protocole LDAP.

```
LDAPMessage ::=
        SEQUENCE {
                messageID           MessageID,
                protocolOp          CHOICE {
                        bindRequest         BindRequest,
                        bindResponse                BindResponse,
                        unbindRequest               UnbindRequest,
                        searchRequest               SearchRequest,
                        searchResponse              SearchResponse,
                        modifyRequest               ModifyRequest,
                        modifyResponse              ModifyResponse,
                        addRequest          AddRequest,
                        addResponse         AddResponse,
                        delRequest                  DelRequest,
                        delResponse         DelResponse,
                        modifyRDNRequest            ModifyRDNRequest,
                        modifyRDNResponse           ModifyRDNResponse,
                        compareDNRequest            CompareRequest,
                        compareDNResponse           CompareResponse,
                        abandonRequest              AbandonRequest
                                                    }
                                }
MessageID ::= INTEGER (0 .. maxInt)
```

Dans la spécification précédente la plupart des noms employés sont directement compréhensibles. A titre d'illustration on ajoute que:

- unbindRequest est le message échangé lors de la fermeture d'un dialogue du client vers le serveur. Il n'y a pas de réponse.

- searchRequest est le message échangé lors de la recherche et/ou de la lecture d'un sous ensemble d'un champs ou d'un attribut de l'annuaire. searchResponse est le message porteur de la réponse.

- abandonRequest est un message généré par un client dans le cas où il souhaite mettre fin à une requête immédiatement sans attendre la réponse. Il n'y a pas de réponse.

4) MessageID est un numéro envoyé par le client dans un message de requête et retourné par le serveur dans une réponse. A quoi sert ce numéro?

Dans le but d'accéder à des informations concernant James Hacker de l'entreprise WidgetInc en Grand Bretage. On souhaite d'abord former la requête LDAP, ensuite on envoie la requête vers le serveur concerné.

Le format PDU d'une requête est comme suit :

BindRequest ::= [APPLICATION 0] SEQUENCE {
version INTEGER (1 .. 127),
name LDAPDN,
authentication AuthenticationChoice }
AuthenticationChoice ::= CHOICE {
simple [0] OCTET STRING, -- 1 and 2 reserved
sasl [3] SaslCredentials, ... }

5) Formez la requête LDAP en indiquant le nom, l'endroit, l'établissement et le pays

6) Quel est l'adressage utilisé par LDAP ? Comment fonctionne l'adressage LDAP ?

7) déterminez l'adresse de l'envoi.

On souhaite maintenant établir l'automate du service client LDAP (pour l'utilisateur du service LDAP) en tenant compte des indications suivantes. On suppose qu'à chaque envoi d'une PDU est associé une primitive de service dont la liste est la suivante (chaque nom est dérivé simplement du nom de la PDU émise):
bind_Req, bind_Resp, unbind_Req, search_Req, search_Resp, modify_Req, modify_Resp, add_Req, add_Resp, del_Req, del_Resp, modifyRDN_Req, modifyRDN_Resp, compare_DNReq, compare_DNResp, abandon_Req

Rappelons que dans un automate de service on identifie toutes les primitives échangées sur l'interface de service d'un logiciel réseau et l'on décrit par un automate uniquement pour ces primitives les enchaînements légaux. Le diagramme état transition utilisé est un automate d'état fini décoré par un ensemble de conditions de franchissement (par exemple associées à la réception de primitives) et d'actions (associées à l'émission de primitives).

8) Établissez l'automate de service client LDAP d'ouverture de connexion (point de vue de l'utilisateur du service LDAP). On suggère pour l'automate de distinguer au moins un état hors connexion, un état connexion établie.

9) On suppose qu'un client doit avoir fini de traiter une requête d'accès à l'annuaire avant de passer à la suivante. Établissez l'automate de service client LDAP pour le traitement d'une requête d'accès à l'annuaire. On pourra prendre pour exemple l'échange search_Req, search_Resp. On se place toujours du point de vue de l'utilisateur du service LDAP. On suggère pour l'automate de distinguer à partir de l'état connexion établie, un état pour le type de requête en cours.

ED7 – XML

XML ('eXtended Markup Language') est un langage de balisage (‘markup langage’), destiné à améliorer l'interopérabilité des échanges de données quelconques entre applications sur l'Internet.

II.1) En dehors de XML, citez différentes normes de définition de structures de données et de formats d’échanges dans les réseaux?

II.2) Qu'est ce qu'un langage de balise? Quelles en sont les utilisations?

II.3) Quel est le principal langage de balise utilisé actuellement?

II.4) Pourquoi définir avec XML encore un langage de balises?

Le document XML suivant définit une bibliographie très simplifiée destinée à être transmise sur Internet pour ensuite être affichée.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<BIBLIOGRAPHIE SUJET="Un exemple">

<LIVRE LANG="fr" SUJET="Informatique">
<AUTEUR> <PRENOM>Christian</PRENOM>
<NOM>Carrez</NOM> </AUTEUR>
<TITRE>Les systèmes informatiques</TITRE>
<EDITEUR> <NOM>Dunod Informatique</NOM>
<LIEU>Paris</LIEU> </EDITEUR>
<DATEPUB>1990</DATEPUB> </LIVRE>

<LIVRE LANG="fr" SUJET="Mathematique">
<AUTEUR> <PRENOM>Norbert</PRENOM>
<NOM>Garsoux</NOM> </AUTEUR>
<TITRE>Espaces topologiques</TITRE>
<EDITEUR> <NOM>Stock</NOM>
<LIEU>Paris</LIEU></EDITEUR>
<DATEPUB>1972</DATEPUB> </LIVRE>

</BIBLIOGRAPHIE>
```

II.5) Dans le document précédent plusieurs éléments qui peuvent posséder plusieurs attributs sont présents. Citez tous les éléments présents et pour chaque élément tous les attributs. Dessinez l’arbre syntaxique associé au document).

Un message électronique Internet (un e-mail à la norme RFC 822) est composé d’une en-tête et d’un corps. L’en-tête contient au minimum les trois informations suivantes (sur trois lignes):

From: expediteur@domaine
To: destinataire@domaine
Date: date_de_creation_du_message

L'entête peut contenir un ensemble de champs optionnels parmi lesquels nous retenons ici:

Subject: sujet du message
Cc: copie@domaine
Message-ID: numero@domaine
Received: information sur le chemin suivi par le message
In-Reply-To: message_ID

Dans une application qui utilise uniquement XML vous avez besoin d'échanger des messages qui comportent exactement les mêmes informations que celles définies précédemment.

II.6) Définissez le document XML qui est équivalent au format d'échange de messages précédent (qui contient les mêmes champs et les mêmes informations).

II.7) Un avantage de XML est de permettre la définition de DTD. Rappelez ce qu'est une DTD et à quoi sert une DTD.

II.8) Définissez la DTD du document XML précédent (associé à un courrier électronique avec ses trois zones obligatoires et ses cinq zones optionnelles).

ED8 – Web Services (première partie)

Les services sur la toile ' **WEB Services '** forment une nouvelle approche de communication de programme à programme entre des applications clientes et des services en ligne sur la toile mondiale (WWW ' World Wide Web '). L'objectif de base des services sur la toile, est donc d'améliorer les communications entre des programmes fonctionnant sur des sites clients distribués dans l'Internet mondial et des applications serveuses. Les domaines d'application visés sont principalement ceux du commerce électronique.

Il s'agit en fait de la définition d'un nouveau type d'intergiciel ('middleware') client serveur qui se place en rival de propositions comme CORBA, DCOM, ou Java RMI. Comme tout intergiciel les services WEB sont définis avec un outil central qui est le protocole de communication et de nombreux outils complémentaires qui traitent des nombreux problèmes annexes (annuaire réparti, sécurité, transactionnel, coordination, …).

Les deux idées essentielles à la base de cette nouvelle approche sont d'une part de faire réaliser les communications entre applications en utilisant le protocole HTTP et d'autre part de spécifier toutes les normes (structures de données, formats de messages) en utilisant le langage XML.

Cette approche est soutenue activement par de très nombreux éditeurs de logiciels. Les entreprises intéressées par les services web et leur processus de normalisation sont réunies au sein de consortiums dont le plus important est le **W3C** ('World Wide Web Consortium').

SOAP ('Simple Object Access Protocol') est le protocole de **communication** des services Web. Il définit deux modes de communications entre programmes : un protocole de communication en mode **message** et un protocole de communication en **mode appel de procédure distante** (**RPC**). SOAP utilise **XML** pour représenter les données échangées. Les normes SOAP sont généralement décrites en supposant l'utilisation de **HTTP** comme protocole sous-jacent (pour acheminer les messages SOAP) mais il est aussi indiqué dans les documents SOAP que des protocoles comme **SMTP** ou d'autres protocoles comme des **MOM** peuvent également être utilisés.

1) SOAP a pour objectif de fournir un mode de communications par messages asynchrones et un mode de communication par RPC. Rappelez les grandes lignes de ces catégories de mode de communication ?

2) Quel est l'usage actuel de HTTP. Soap propose d'utiliser HTTP comme un moyen de communications entre programmes. Quels avantages et quels inconvénients voyez vous à la réalisation de communications entre programmes en utilisant le protocole HTTP ?

3) On rappelle que le protocole HTTP est présenté dans ses RFC comme offrant une communication au niveau des objets. HTTP utilise dans son vocabulaire la notion de méthodes. Quelles sont les différentes méthodes offertes en http ?

4) HTTP vous semble t'il correspondre à un protocole d'appel de procédure distante (justifiez votre réponse) ?

5) Les concepteurs de SOAP ont choisi XML comme outil de représentation de la structure des messages. Quel est le principal langage, utilisé avant XML, pour représenter les différents types de données manipulés par des programmes et échangés par messages ?

6) Quels sont les avantages et quels sont les inconvénients de l'utilisation de XML pour représenter la structure des messages échangés dans un protocole comme SOAP ?

Dans le fonctionnement de SOAP en mode RPC, voici un exemple de message d'appel (extrait du document W3C SOAP primer). L'objectif de l'échange est de confirmer une réservation précédemment faite et d'effectuer le paiement.

```
POST /Reservations HTTP/1.1
Host: travelcompany.example.org
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
  <env:Header>
  </env:Header>
  <env:Body>
    <m:chargeReservation
      env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
      xmlns:m="http://travelcompany.example.org/">
      <m:reservation xmlns:m="http://travelcompany.example.org/reservation">
      <m:code>FT35ZBQ</m:code>
      </m:reservation>
      <o:creditCard xmlns:o="http://mycompany.example.com/financial">
      <n:name xmlns:n="http://mycompany.example.com/employees">
        Åke Jógvan Øyvind
      </n:name>
      <o:number>123456789099999</o:number>
      <o:expiration>2005-02</o:expiration>
      </o:creditCard>
    </m:chargeReservation>
  </env:Body>
</env:Envelope>
```

Le message résultat correspondant au message de requête précédent a la forme suivante (la plupart des détails sont omis pour limiter le volume de l'exemple) :

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
 <env:Header>
  ...
  ...
 </env:Header>
 <env:Body>
  ...
  ...
 </env:Body>
```

```
</env:Envelope>
```

7) A quoi correspondent les cinq premières lignes du message d'appel. Commentez les informations qui se trouvent dans chacune des cinq lignes ?

8) Le document XML qui encode l'appel est composé d'une entête vide et d'un corps qui contient le nom de la méthode à exécuter et les paramètres d'appel. Quel est l'arbre résultant de l'analyse syntaxique du document précédent ?

L'exemple suivant montre un fonctionnement de SOAP avec SMTP. Le message de requête est alors le suivant:

```
From: a.oyvind@mycompany.example.com
To: reservations@travelcompany.example.org
Subject: Travel to LA
Date: Thu, 29 Nov 2001 13:20:00 EST
Message-Id: <EE492E16A090090276D208424960C0C@mycompany.example.com>
Content-Type: application/soap+xml

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    ...
  </env:Header>
  <env:Body>
    ...
  </env:Body>
</env:Envelope>
```

9) Expliquez comment fonctionne le protocole soap avec SMTP (est ce que reservations@travelcompany.example.org est le nom du programme qui exécute l'appel) ?

10) Quel(s) avantage(s) peut-on trouver à l'utilisation de SMTP comme protocole de transport des messages SOAP ?

11) WSDL ('Web Service Definition Language') est le langage de définition d'interfaces des services web (IDL 'Interface Definition Language'). Rappelez le rôle d'un tel langage dans le cadre d'un protocole de communication en approche objets répartis (par exemple dans le standard CORBA) ?

Comme tous les outils définis dans le cadre des services sur la toile, WSDL est un dialecte XML. La structure d'une déclaration WSDL est décrite dans ses grandes lignes comme suit.

WSDL commence par permette à un utilisateur de décrire de nouveaux types de données dans un langage baptisé xmlschéma (cette possibilité est décrite par les trois lignes suivante dans la description générale de la structure d'un document WSDL)

```
<wsdl:types>
    <xsd:schema> ...</xsd:schema>
</wsdl:types>
```

WSDL propose ensuite de décrire la structure d'un ensemble de messages qui sont en fait des messages de requêtes et de réponse utilisés dans le cadre de RPC.

```
<wsdl:message name="nmtoken">
```

```
    <part name="nmtoken" element="qname"? type="qname"?/>
  </wsdl:message>
```

Ensuite WSDL définit un élément type de port comme un ensemble d'opérations. Une opération est définie par un message d'appel (input) et un message résultat (output). Le standard prévoit aussi la description d'une structure de données associée aux erreurs.

```
  <wsdl:portType name="nmtoken">
    <wsdl:operation name="nmtoken">
      <wsdl:input name="nmtoken"? message="qname">
      </wsdl:input>
      <wsdl:output name="nmtoken"? message="qname">?
      </wsdl:output>
      <wsdl:fault name="nmtoken" message="qname">
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:portType>
```

L'élément binding (liaison) définit ensuite concrètement comment accéder à un service en indiquant son type de port, le protocole utilisé pour y accéder (c'est principalement Soap en mode HTTP mais on a vu que d'autres possibilités sont prévues) et la localisation dans la toile de ce service (URI de l'endroit ou se trouve le service web).

```
  <wsdl:binding name="nmtoken" type="qname">
    <wsdl:operation name="nmtoken">
      <wsdl:input>
      </wsdl:input>
      <wsdl:output>
      </wsdl:output>
      <wsdl:fault name="nmtoken">
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
```

Finalement le service rendu sur le Web par une entreprise peut être composé de plusieurs ports c'est à dire de plusieurs applications accessibles sur le web (correspondant à des liaisons ou 'binding' que nous venons de voir) et donc localisées éventuellement sur des serveurs différents.

```
    <wsdl:service name="nmtoken">
      <wsdl:port name="nmtoken" binding="qname">
      </wsdl:port>
    </wsdl:service>
```

12) Comparez ce langage de définition d'interface à celui de Corba.
Suggestions pour construire la réponse :
Pourquoi décrire des types de données (à quoi correspond cette possibilité en IDL CORBA) ?

Pourquoi décrire des types de ports qui regroupent des opérations (à quoi correspond cette possibilité en IDL CORBA) ?

Pourquoi décrire des ports qui définissent des points d'accès concrets à des services (à quoi correspond cette possibilité en IDL CORBA).

ED9 – Web Services (deuxième partie)

Conception d'une application Web

On souhaite développer une application de commerce électronique qui permet d'obtenir l'heure de la part d'un serveur distant. A cette fin, la méthode getTime a été développée. Elle prend en entrée le fuseau horaire de façon à fournir en sortie la date. Il vous est demandé de développer cette application de type client-serveur en utilisant XML-RPC.

1/ Rappeller brièvement les principes généraux de XML-RPC.

Aucune spécification officielle de XML-RPC n'a été développée. Il vous est demandé de développer une DTD d'une requête XML-RPC. Pour rappel, un appel à procédure distante (élément racine désigné par l'étiquette <methodCall>) contient le nom de la procédure invoquée (étiquette <methodName>). Ce nom est un identifiant correspondant à une chaîne de caractères (caractère minuscule ou majuscule a-z, A-Z, caractères numériques 0-9, auxquels s'ajoutent les caractères spéciaux underscore, point, deux points et slashe). Ce nom est suivi par l'étiquette <params> qui regroupe l'ensemble des paramètres d'entrée du XML-RPC(étiquette <param>), chacun de ces derniers ayant une valeur (étiquette <value>). Une valeur peut correspondre aux types primitifs des RPC-XML, à une structure (<struct>) ou un tableau (<array>).

2) Donner la DTD d'un message RPC-XML

3/ Quels sont les lacunes des DTD (en terme d'expressivité) ?

4/ Donner le schémas XML des éléments correspondants aux types primitifs d'une requête XMLRPC.
A ces types primitifs s'ajoutent deux types plus complexes, la structure et le tableau :

```
<xsd:complexType name="StructType">
<xsd:sequence>
<xsd:element name="member" type="MemberType"
maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="MemberType">
<xsd:sequence>
<xsd:element name="name" type="xsd:string" />
<xsd:element name="value" type="ValueType"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ArrayType">
<xsd:sequence>
<xsd:element name="data">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="value" type="ValueType"
minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
```

```
</xsd:complexType>
```

5/ A partir des éléments définis précédemment, présentez le shémas XML d'une valeur (ValueType) d'un paramètre (ParamType)

6/ Donner un schémas XML d'une requête RPC, sachant que le nom d'une méthode est défini de la façon suivante :

```
<xsd:simpleType name="nameType">
<xsd:restriction base="xsd:tocken">
<xsd:pattern value="([A-Za-z0-9]|/|\.|:|_)*" />
</xsd:restriction>
</xsd:simpleType>
```

7/ Donnez le message requête XML-RPC qui serait envoyé par un client pour connaître la date étant donné le fuseau horaire GMT. Suivant la norme Iso 8601, 23heures 22 minutes 10 secondes le 22/07/1998 serait représenté 1998-07-22T23:22:10.

8/ Au regard des DTD et schémas XML que vous venez de définir, quelles sont les lacunes de XML-RPC? Quelles solutions pour pallier à ces problèmes?

9/ Au regard des limitations de XML-RPC, il vous est demandé d'implémenter cette application en tant que service WEB. Le document XML présenté ci-dessous présente une requête SOAP.

Décrivez le format de la requête ci-dessous.

```
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle=
"http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:getTime xmlns:ns1="urn:MySoapServices">
< fuseauHoraire
xsi:type="xsd:string">GMT</fuseauHoraire>
</ns1:getTime>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10/ Vous parait-il intéressant de développer cette application Web en vous basant sur une architecture de type REST?

11/ Le service Web que vous avez implémenté pourrait-il est envisagé pour remplacer un protocole de type NTP (Network Time Protocol)?

ED10 – ASN1

Exercice 1 : Définition de syntaxe ASN1

On donne la spécification générale en ASN1 de la zone de donnée utilisateur d'une PPDU (Presentation Protocol Data Unit, unité de protocole de présentation). Dans cette spécification PDV est l'abbréviation de Presentation Data Value. L'exercice consiste à commenter cette spécification pour comprendre les choix effectués et par suite préparer correctement des messages.

```
User-data ::= CHOICE
        {
        [APPLICATION 0]
            IMPLICIT Simply-encoded-data.
        [APPLICATION 1]
            IMPLICIT Fully-encoded-data.
        }
Simply-encoded-data ::=
    OCTET STRING
Fully-encoded-data ::=
    SEQUENCE OF
        PDV-list
PDV-list ::=
    SEQUENCE
        {
        Transfer-syntax-name OPTIONAL,
        Presentation-context-identifier,
        presentation-data-values
        CHOICE {
            single-ASN1-type[0]
                ANY,
            octet-aligned[1]
                IMPLICIT OCTET STRING,
            arbitrary[2]
                IMPLICIT BIT STRING
                }
        }
Presentation-context-identifier ::= INTEGER
Transfer-syntax-name ::= OBJECT IDENTIFIER
```

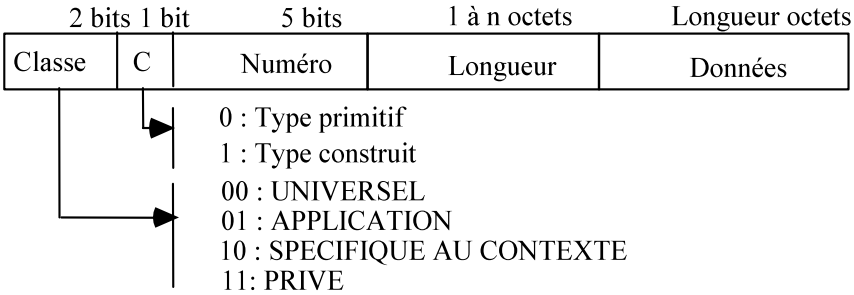
1 Le concepteur a défini deux formats principaux de données utilisateurs. Quels sont-ils? En fonction de leurs définitions a quels besoins correspondent t'ils selon vous?
2 La définition de la syntaxe de transfert utilisée est optionnelle. Pourquoi ce choix est-il effectué?
3 La définition du contexte de présentation vient ensuite. A quoi correspond cette notion? Pourquoi est-elle obligatoire?
4 Dans les différentes possibilités de PDV on trouve trois formats. Quels sont-ils? A quels besoins correspondent-ils selon vous?

Exercice 2 : Définition de format de transfert ASN1

On souhaite pour une application graphique 2D échanger des objets dont l'un des types est le point à coordonnées entières. Un point va donc être défini comme un type séquence comportant deux entiers, d'étiquette application et pour gagner un peu de place on va éviter de renvoyer le type séquence en le déclarant implicite.

Question 1 - Donnez la déclaration en syntaxe abstraite du point? Rappelez la signification précise dans cette déclaration les mots clés APPLICATION, IMPLICIT, SEQUENCE?

Question 2 - On rappelle que la syntaxe de transfert ASN1 repose sur la structure suivante. Le numéro associé au type universel entier est 2. On suppose que tous les objets sont définis par leur longueur explicitement donnée dans les champs longueur (pas par un délimiteur de fin).



Donnez en hexadécimal le codage de transfert d'un point de coordonnées (5,4)

Exercice 3 : Définition de données en ASN1 et de leur format de transfert.

On souhaite pour une application de gestion accéder à distance à différents items de données dont la clé est le numéro national d'identité.

On donne la définition suivante du numéro national d'identité qui est composé de six champs (définition simplifiée) :

- le champ sexe est un entier qui vaut 1 ou 2,
- l'année de naissance est un entier sur deux chiffres décimaux compris entre 00 et 99,
- le mois de naissance est un entier sur deux chiffres décimaux compris entre 01 et 12,
- le numéro de département de naissance est un entier sur deux chiffres décimaux ,
- le numéro de la commune de naissance dans le département de naissance est un entier sur trois chiffres décimaux,
- le numéro de la personne sur le registre de l'état civil est également un entier sur trois chiffres décimaux.

Question 1 - Définissez en syntaxe abstraite ASN1 le numéro national d'identité. Différentes solutions sont possibles. On pourra s'aider des exemples donnés en cours ou en exercices en expliquant la nature des choix effectués. .

Question 2 Le numéro associé au type universel construit SEQUENCE ou SEQUENCE OF est 16, et celui du type universel primitif ENTIER est 2, le type NumericString est codé 18 . On suppose que tous les objets sont définis par leur longueur explicite (pas par un délimiteur de fin).

Donnez le codage de transfert d'une personne de numéro 2510641008009 soit en syntaxe ASN1 l'instance suivante du type défini à la question précédente: { 2, 51, 06, 41, 008 , 009 } (en hexadécimal).

Exercice 4 : Problème général ASN1

On souhaite construire un serveur d'annuaire téléphonique simple. Ce serveur d'annuaire peut être consulté et mis a jour par des utilisateurs. Nous avons donc un modèle client serveur.

Question 1 - Donner une structure de données en C pour un enregistrement du fichier correspondant à l'annuaire.
Question 2 - Donner la structure ASN1 et donner une idée du codage.
Question 3 - Définir les opérations possibles sur ce fichier.
Question 4 - Définir le protocole d'accès au serveur (format des requêtes, des réponses et des erreurs)
Question 5 - Coder ce protocole en ASN1.

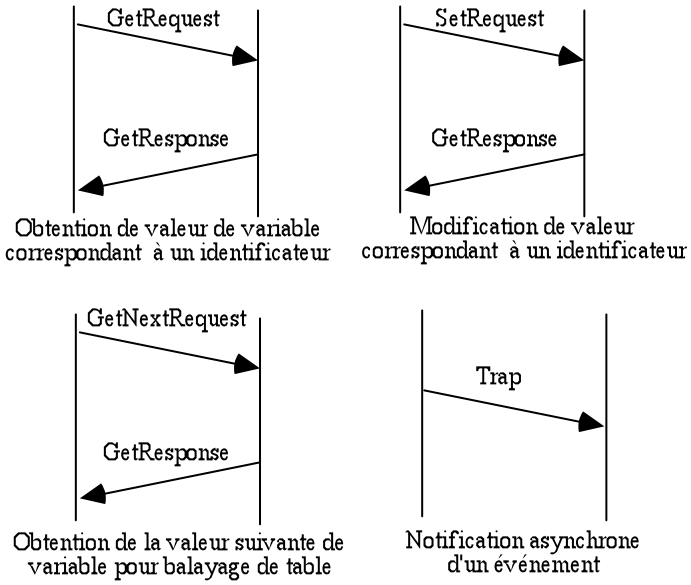
ED11-SNMP

Administration de réseaux avec SNMP

Le protocole SNMP (Simple Network Management Protocol) est un protocole d'administration de réseaux développé dans le contexte du réseau INTERNET. Son objectif principal est de permettre décrire des applications (baptisée station d'administration SNMP) qui vont d'acquérir en mode client/serveur des valeurs caractéristiques du fonctionnement d'un appareil distant. Cet entité réseau est baptisé agent SNMP. Les paramètres lus ou positionnés sur l'agent SNMP sont appelés variables ou objets dans la suite.

Certaines variables associées au fonctionnement de base de l'INTERNET sont prédéfinies et typées selon des RFC (Request For Comments) pour former les définitions de structures (SMI: Structure of Management Information). Il s'agit, par exemple, des adresses IP (IPadress) ou des durées mesurées en TimeTicks de 10 millisecondes. Elles sont rangées dans des bases de données d'administration ou MIB sous forme de tables ou relations par des agents SNMP de façon à être lues par des applications SNMP.

Les échanges de PDU SNMP s'opèrent selon quatre modes définis par les diagrammes suivants:



La RFC 1157 donne la définition suivante d'un message SNMP qui utilise la syntaxe abstraite ASN1.

```
RFC1157-SNMP DEFINITIONS ::= BEGIN
IMPORTS
        ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
                FROM RFC1155-SMI;
Message ::= SEQUENCE {
                version INTEGER  {version-1 (0)}, -- Version 1 for this RFC
                community OCTET STRING,                 -- Community name
                data ANY                                -- e.g. PDUs
```

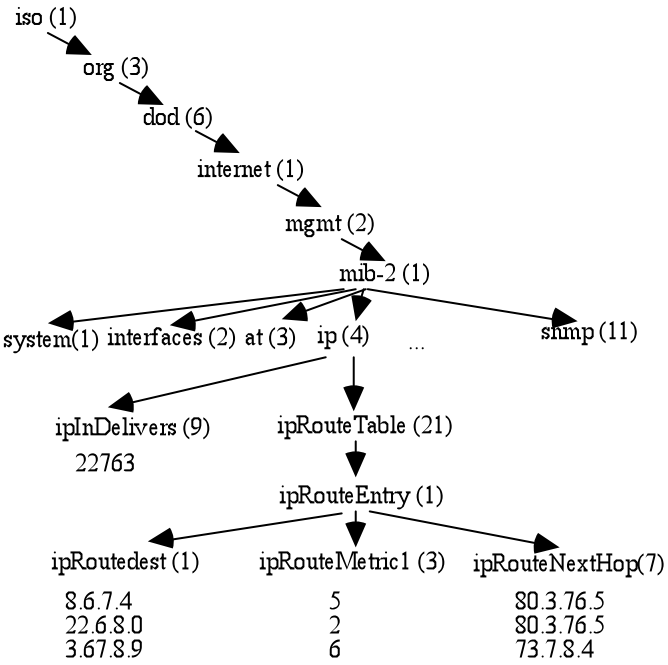
```
                    }
PDUs ::= CHOICE {  -- Protocol data units
                get-request             GetRequest-PDU,
                get-next-request        GetNextRequest-PDU,
                get-response            GetResponse-PDU,
                set-request             SetRequest-PDU,
                trap                    Trap-PDU
                }
GetRequest-PDU      ::= [0] IMPLICIT PDU
GetNextRequest-PDU::=[1] IMPLICIT PDU
GetResponse-PDU    ::= [2] IMPLICIT PDU
SetRequest-PDU      ::= [3] IMPLICIT PDU
PDU ::= SEQUENCE {
                request-id INTEGER,          -- Request identifier
                error-status INTEGER {              -- Sometimes ignored
                        noError (0),
                        toobig (1),
                        noSuchName (2),
                        badValue (3),
                        readOnly (4),
                        genError (5)},
                error-index INTEGER,                -- Sometimes ignored
                                                    -- Index of the faulty variable
                variable-binding VarBindList }      -- Values are sometimes ignored
Trap-PDU ::= [4] IMPLICIT SEQUENCE {
                enterprise      OBJECT IDENTIFIER,      -- Type of object generating trap
                agent-addr      NetworkAddress,         -- Only one type of network adresses
                                                        -- IP adress of object generating trap
                generic-trap    INTEGER {               -- Generic trap type
                        coldStart (0),
                        warmStart ( 1),
                        linkDown ( 2),
                        linkUp (3),
                        authenticationFailure (4),
                        egpNeighborLoss ( 5),
                        enterpriseSpecific (6)
                        },
                specific-trap   INTEGER,   -- Specific code
                time-stamp TimeTicks,   -- Elapse time since the last reinitialization of the entity
                variable-binding VarBindList  -- "Interesting" information
                }
VarBind ::= SEQUENCE      - Variable binding
                {name ObjectName,
                value ObjectSyntax}
VarBindList ::= SEQUENCE OF VarBind
END
```

La désignation des objets d'administration SNMP se fait selon une méthode de construction arborescente des noms où chaque nœud peut être repéré par une chaîne de caractères ou un code

numérique entier. Pour décrire un nœud de l'arbre on utilise une notation pointée pour les noms logiques de l'Internet.

Dans le schéma suivant on ne donne qu'une vue très partielle de l'arbre de désignation. On représente la branche associée à l'Internet et on s'intéresse au protocole IP.

Pour celui-ci on fait apparaître l'information du compteur entier du nombre de paquets correctement délivrés (ipInDelivers) dont la valeur est 22763. Ensuite on détaille quelques éléments de routage. La table de routage (ipRouteTable) comprenant pour chaque entrée (ipRouteEntry) différentes informations. Nous avons gardé l'adresse d'un destinataire à atteindre dans l'Internet (ipRoutedest). L'adresse est au format IP sur 32 bits représentés par 4 chiffres décimaux séparés par des points. Le coût du chemin pour atteindre ce destinataire (ipRouteMetric1) est un entier. L'adresse du prochain site à visiter dans l'Internet (ipRouteNextHop) pour rejoindre le destinataire est une adresse IP. On a représenté à titre d'exemple académique les valeurs d'une table de routage qui ne comporte que trois entrées.



La désignation absolue de l'objet ipInDelivers est:

iso.org.dod.internet.mgmt.mib-2.ip.ipInDelivers soit 1.3.6.1.2.1.4.9.

La valeur de la variable ipInDelivers est obtenue par le message
GetRequest (1.3.6.1.2.1.4.9) qui produit la réponse
GetResponse ((ipInDelivers = 22763))

On souhaite maintenant consulter la table de routage par la commande GetNextRequest .
GetNextRequest (ipRoutedest, ipRouteMetric1,ipRouteNextHop)
On obtient alors la réponse suivante (qui correspond à une entrée de la table suivant l'entrée courante en l'occurrence la première):
GetResponse ((ipRoutedest.22.6.8.0="22.6.8.0"),
(ipRouteMetric1.22.6.8.0="2"),
(ipRouteNextHop 22.6.8.0="80.3.76.5"))
On continue la consultation de la table par :
GetNextRequest (ipRoutedest.22.6.8.0, ipRouteMetric1.22.6.8.0, ipRouteNextHop.22.6.8.0)
On obtient la réponse (l'entrée suivante de la table):
GetResponse ((ipRoutedest.3.67.8.9="3.67.8.9"),

(ipRouteMetric1.3.67.8.9="6"),
(ipRouteNextHop.3.67.8.9="73.7.8.4"))

Enfin, par la dernière requête de la consultation de la table on obtient la dernière réponse qui correspond à la première ligne de la table:
GetNextRequest (ipRoutedest.3.67.8.9, ipRouteMetric1.3.67.8.9, ipRouteNextHop.3.67.8.9)

on obtient la réponse suivante :
GetResponse ((ipRoutedest.8.6.7.4="8.6.7.4"),
(ipRouteMetric1.8.6.7.4="5")
(ipRouteNextHop.8.6.7.4="80.3.76.5"))

Une nouvelle requête GetNextRequest sur une table entièrement parcourue produit une réponse particulière permettant de tester la fin. On désigne ici pour simplifier l'énoncé un tel code réponse par un booléen Fin-de-table positionné automatiquement.

I Questions ASN1

I.1. La spécification du message GetRequest-PDU comporte une directive **[0] IMPLICIT**. Que signifie-t-elle?
I.2. Dans la spécification ASN.1 dont vous disposez, quel est le type qui décrit les objets administrés par les agents SNMP ?
I.3. A quoi correspondent les différents champs du type PDU?
I.4. A la vue des commentaires de la description du type PDU, réécrivez ce type en ASN.1 en cherchant à optimiser l'encombrement des messages (certains champs ne sont pas toujours obligatoires) ? Justifiez votre réécriture

II Question d'organisation architecturale

II.1. Le protocole SNMP utilise la couche transport UDP de l'INTERNET pour acheminer ses messages. Donnez les avantages et les inconvénients de l'utilisation de cette couche de transport. Cela vous semble-t-il être un bon choix dans le cas de SNMP?

II.2. Un message SNMP peut-être compressé pour des raisons d'efficacité et crypté pour des raisons de confidentialité. On rappelle que sa définition ASN1 permet la conversion des informations pour tout type de matériel. On est ainsi placé devant différents choix possibles concernant l'organisation des trois fonctions pouvant s'appliquer à un message. Dans quel ordre placez-vous les trois opérations de compression, conversion et chiffrement? Justifiez très précisément votre choix.

III Questions SNMP

III.1. Quelle est la désignation codée numérique de l'objet ipRouteMetric1?

III.2. Donnez des exemples d'usages possibles de la PDU trap par un agent SNMP autres que ceux donnés dans la description de base?

IV Questions de spécifications d'applications

IV.1 On souhaite sur requête d'un opérateur (?affiche-table-routage) déclencher l'affichage d'une table de routage telle que celle définie par ipRouteTable. Donnez l'automate d'une application SNMP de lecture et d'édition de la table de routage?

IV.2 On souhaite enregistrer dans un fichier pour chaque intervalle de 100 millisecondes le nombre de paquets délivrés correctement. On veut noter 10000 échantillons pour des statistiques ou des simulations ultérieures. On dispose d'une horloge interne qui peut délivrer des événements au bout de 100 millisecondes à condition d'avoir été armée. L'événement retombée du signal d'horloge est alors tout à fait analogue à l'arrivée d'un message. Donnez l'automate permettant de créer ce fichier?

ED12 - Étude du protocole HTTP

Le but de ce problème est d'étudier le protocole HTTP ("Hyper Text Transfer Protocol") utilisé pour implanter le service d'accès à des données hypertextes Web de l'Internet. Il utilise les services offerts par la suite des protocoles TCP/IP.

1) Le résumé qui définit ce protocole indique: "HTTP est un protocole de la couche application, sans état, construit pour utiliser des systèmes multimédia et distribués. Il implante un service client/serveur entre programmes".

1.1) Que signifie le qualificatif "sans état"?

1.2) Parmi les divers protocoles étudiés en cours donnez un exemple d'un autre protocole sans état?

2) Il est aussi précisé dans la définition d'HTTP :
"Les messages sont échangés dans un format similaire à celui utilisé par le courrier Internet et son extension MIME "Multipurpose Internet Mail Extensions".

2.1) Quel est le rôle joué par MIME?

2.2) Quelle est la couche de la pile des protocoles ISO/ISO qui joue le même rôle?

3) Pour décrire les données manipulées par HTTP, on utilise des définitions syntaxiques BNF ("Backus Naur Form"). Cette syntaxe permet de définir des identifiants bien formés (comme on va le voir pour les adresses de ressources) à l'aide de règles BNF de la forme suivante:

nom = définition

Le nom à construire est obtenu à l'aide de la définition dans laquelle interviennent des symboles non-terminaux (mots à remplacer par leur définition) et des symboles terminaux (mots à écrire en toute lettre) écrits entre guillemets (par exemple "ABC" désigne la chaîne de caractères ABC. On utilise les conventions suivantes:

règle1 règle2	**Signifie que règle1 suivie de règle2 s'applique**
[regle]	**Signifie que cette règle est optionnelle.**
***(règle)**	**Signifie que la règle apparaît 0 ou un nombre quelconque de fois.**
1*(règle)	**Signifie que la règle apparaît une fois ou un nombre quelconque de fois.**
règle1 | règle2	**Signifie que règle1 ou règle2 s'applique (le ou étant exclusif)**

Les ressources réseaux atteintes par le protocole HTTP sont repérées par une adresse appelée URL ("Uniform Ressource Locator"). La syntaxe (simplifiée) d'une URL HTTP est définie sous forme BNF par:

```
http_URL = "http:" "//" service [":" port ] [ abs_path ]
service = un_nom_de_service_Internet
port = 1*DIGIT
abs_path = "/" [ path ]
path = fsegment *("/" segment)
fsegment = 1*(suitedecaractères)
```

segment = *(suitedecaractères)
DIGIT = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
suitedecaractères = A..Z|a..z|0..9|.|_|-

On ne définit pas la syntaxe des noms de services Internet (symbole un_nom_de_service_Internet).

Sachant que www.inria.fr est un un_nom_de_service_Internet valide indiquez parmi les URL HTTP ci dessous, celles qui sont valides et celles qui ne le sont pas en justifiant vos réponses.

3.1) http://www.inria.fr
3.2) http://www.inria.fr/
3.3) http://www.inria.fr:/tech-reports/fich.html
3.4) http://www.inria.fr:8080/pub/logiciel

4) Une requête http consiste à demander au serveur l'exécution d'une méthode (i.e. d'une action) sur une ressource.
La méthode GET utilisée par un client est une requête demandant le contenu se trouvant à URL_de_l_objet.

Par exemple une requête comme:
GET http://www.inria.fr/pub/courrier/ HTTP/1.0
demande au serveur www.inria.fr de renvoyer le document /pub/courrier.

Un tel document est en général une page statique déposée au niveau du serveur.

La méthode POST permet de soumettre des interrogations à une base de données locale au serveur.

Une syntaxe simplifiée d'une requête http est donnée par les règles suivantes:

Requête = Méthode SP URL_de_l_objet SP version_protocole_http
Méthode = "POST" | "GET"
SP = 1*(" ")
URL_de_l_objet = http_URL
version_protocole_http = "HTTP/1.0"

Les requêtes ci dessous sont-elles valides ou invalides :
4.1) OPTIONS * HTTP/1.0
4.2) GET http://www.inria.fr:8080/pub/logiciel HTTP/1.0
4.3) GET http://www.inria.fr/pub/courrier/ HTTP/1.0
4.4) POST GET http://www.inria.fr HTTP/1.0

5) Que répond un serveur dans le cas d'une requête non valide?

6) HTTP est un protocole client/serveur et, donc, se pose le problème de l'idempotence des requêtes.
6.1) Rappelez la définition de la notion d'idempotence

6.2) Donnez des exemples de méthodes idempotentes et de méthodes qui ne le sont pas

6.3) Est ce que les méthodes GET et POST sont idempotentes ?

7) Une implantation d'un serveur HTTP peut utiliser la notion de proxy ou mandataire. Un proxy est un programme pouvant jouer le rôle de client comme celui de serveur. Il peut recevoir une requête et la traiter directement. Il peut également déléguer une requête à un autre serveur en la reformattant.
7.1) Donnez plusieurs utilisations possibles d'un proxy

7.2) Indiquez les avantages et les inconvénients de l'emploi d'un proxy.

8) Une page HTML peut contenir du texte HTML, des images fixes ou animées, du son, des scripts CGI et des applets Java. Seuls le texte HTML, les images fixes (par exemple le format d'image GIF), le résultat de scripts CGI et les applets Java sont chargés lors de la demande de consultation d'une page par un client, les autres étant chargés à la demande. Donc, pour chaque page HTML, le client envoie une première requête qui permet le chargement du texte HTML. Lors de la réception du texte de la page, il analyse le contenu et demande, ensuite, le chargement des éléments nécessaires par d'autres requêtes GET au serveur. On considère une page HTML contenant 4 images au format GIF, un fichier son, 1 applet Java, un script CGI dont le résultat est une image au format GIF.
8.1) Quel est le nombre de requêtes GET émises par le client vers le serveur?

8.2) Quel est le choix effectué pour la transmission des séquences vidéo?

9) Les balises HTML suivantes délimitent:
- **une image fixe : <IMG SRC = "...">,**
- **une applet Java : <APPLET CODE = "...">**
- **le résultat d'un script affichable <IMG SRC = "...">.**

La commande indiquant l'URL chez le client est:
***charger URL = http_URL*.**
On souhaite définir l'automate du chargement d'une page à partir de la commande *charger* émise par un client (comportement du navigateur)
9.1) Donnez la liste des états que vous identifiez dans un tel automate ?
9.2) Donnez l'automate du client complet en le commentant
Vous utiliserez les requêtes et réponses appropriées définies dans le protocole HTTP et traiterez éventuellement les cas d'erreur.

10) Le client utilise maintenant un cache local "client". Ce cache permet de garder des pages déjà consultées pendant un certain temps (défini par l'utilisateur). La place disponible sur disque pour ce cache client étant limitée, les pages les plus anciennes sont remplacées au fur et à mesure. Quels sont les avantages et les inconvénients de l'utilisation d'un cache client ?

11) On rappelle que, pour tenir compte du vieillissement d'une page, chaque page consultée contient un champ d'en-tête "date d'expiration". Par définition la date

d'expiration du résultat d'un script CGI est toujours dépassée. Modifiez l'automate précédent pour prendre en compte l'utilisation du cache local du client.

12) L'utilisateur a modifié la configuration de son client navigateur de manière à ne plus utiliser de cache local client mais un cache serveur sur son site.
12.1) Comment se comporte maintenant un client?

12.2) Quel est l'interêt de cette nouvelle organisation?