

**Sujets des exercices dirigés**

**Technologie pour les applications  
client-serveur**

**UE RSX 102**

**Année 2007-2008**

**G. Florin, S. Natkin, E. Gressier,  
L. Duchien, J.M. Farinone**

# Table des matières

<b>PREMIER CHAPITRE.....</b>	<b>4</b>
<b>EXERCICES CONTROLE REPARTI.....</b>	<b>4</b>
SERIE D'EXERCICES MODE MESSAGE.....	4
<i>Exercice 1 : Automate de connexion TCP.....</i>	4
SERIE D'EXERCICES : APPEL DE PROCEDURE DISTANTE.....	6
<i>Exercice 2 : Traitement des pertes de messages.....</i>	6
<i>Exercice 3 : Exécutions très longues.....</i>	6
<i>Exercice 4 : Appels en parallèle (redondance massive).....</i>	6
<i>Exercice 5 : Serveurs multiples (redondance sélective).....</i>	6
<i>Exercice 6 : Parallélisme et performance des communications en RPC.....</i>	7
SERIE D'EXERCICES : ETUDES DE CAS.....	9
<i>Exercice 7 : Conception d'une application de niveau session.....</i>	9
<i>Exercice 8 : Comparaison de la programmation d'applications réparties avec TCP et avec CORBA.....</i>	11
<i>Exercice 9 : Objets répartis en CORBA.....</i>	16
<b>SECOND CHAPITRE.....</b>	<b>18</b>
<b>EXERCICES PRÉSENTATION DES DONNEES .....</b>	<b>18</b>
SERIE D'EXERCICES CONVERSIONS.....	18
<i>Exercice 1 : Optimisation des conversions.....</i>	18
<i>Exercice 2 : Problème des formats de mémoire (petit boutiste, grand boutiste).....</i>	18
<i>Exercice 3 : Définition de syntaxe ASN1.....</i>	19
<i>Exercice 4 : Définition de format de transfert ASN1.....</i>	20
<i>Exercice 5 : Définition de données en ASN1 et de leur format de transfert.....</i>	20
<i>Exercice 6 : Problème général ASN1.....</i>	21
<i>Exercice 7 : Définition d'une traduction du langage IDL de CORBA vers le langage Pascal.....</i>	22
<i>Exercice 8 : CORBA.....</i>	24
SERIE D'EXERCICES SECURITE.....	29
<i>Exercice 9 : Cryptogramme 1.....</i>	29
<i>Exercice 10 : Cryptogramme n° 2.....</i>	29
<i>Exercice 11 : Cryptogramme n° 3.....</i>	29
<i>Exercice 12 : Etude du système de chiffrement à clé publique RSA.....</i>	30
<i>Exercice 13 : Kerberos.....</i>	31
<i>Exercice 14 : Partage d'un secret.....</i>	34
<i>Exercice 15 : Problème de notarisation.....</i>	34
<i>Exercice 16 : Gestion d'une connexion sécurisée.....</i>	36
<i>Exercice 17 : Changement périodique de clés en cryptographie à clés publiques.....</i>	38
<i>Exercice 18 : Commerce électronique sur Internet: SET ("Secure Electronic Transactions").....</i>	41
<i>Exercice 19 : Authentification des usagers et autorisation des requêtes dans le WEB.....</i>	45
<i>Exercice 21 : Sécurisation des communications en Internet avec SSL-TLS.....</i>	49
<i>Exercice 22 : Protocole de micro-paiement sur Internet.....</i>	52
<i>Exercice 23 : Sécurisation du DNS : les normes DNSSEC-1.....</i>	55
<b>TROISIEME CHAPITRE.....</b>	<b>59</b>
<b>EXERCICES APPLICATIONS REPARTIES.....</b>	<b>59</b>
<i>Exercice 1 : Système de fichiers répartis NFS et réplication.....</i>	59
<i>Exercice 2 : Désignation des fichiers dans le système de fichiers répartis NFS.....</i>	60
<i>Exercice 3 : Messagerie X400.....</i>	62
<i>Exercice 4 : Transactionnel réparti OSI-TP.....</i>	64
<i>Exercice 5 : Transactionnel réparti.....</i>	65
<i>Exercice 6 : Système transactionnel réparti.....</i>	66
<i>Exercice 7 : CORBA et le transactionnel.....</i>	68
<i>Exercice 8 : Administration de réseaux avec SNMP.....</i>	70
<i>Exercice 9 : Échange de données informatisées EDI.....</i>	75
<i>Exercice 10 : XML.....</i>	77
<i>Exercice 11 : Étude du protocole HTTP.....</i>	79
<i>Exercice 12 : DNS ('Domain Name System') sujet 1.....</i>	82

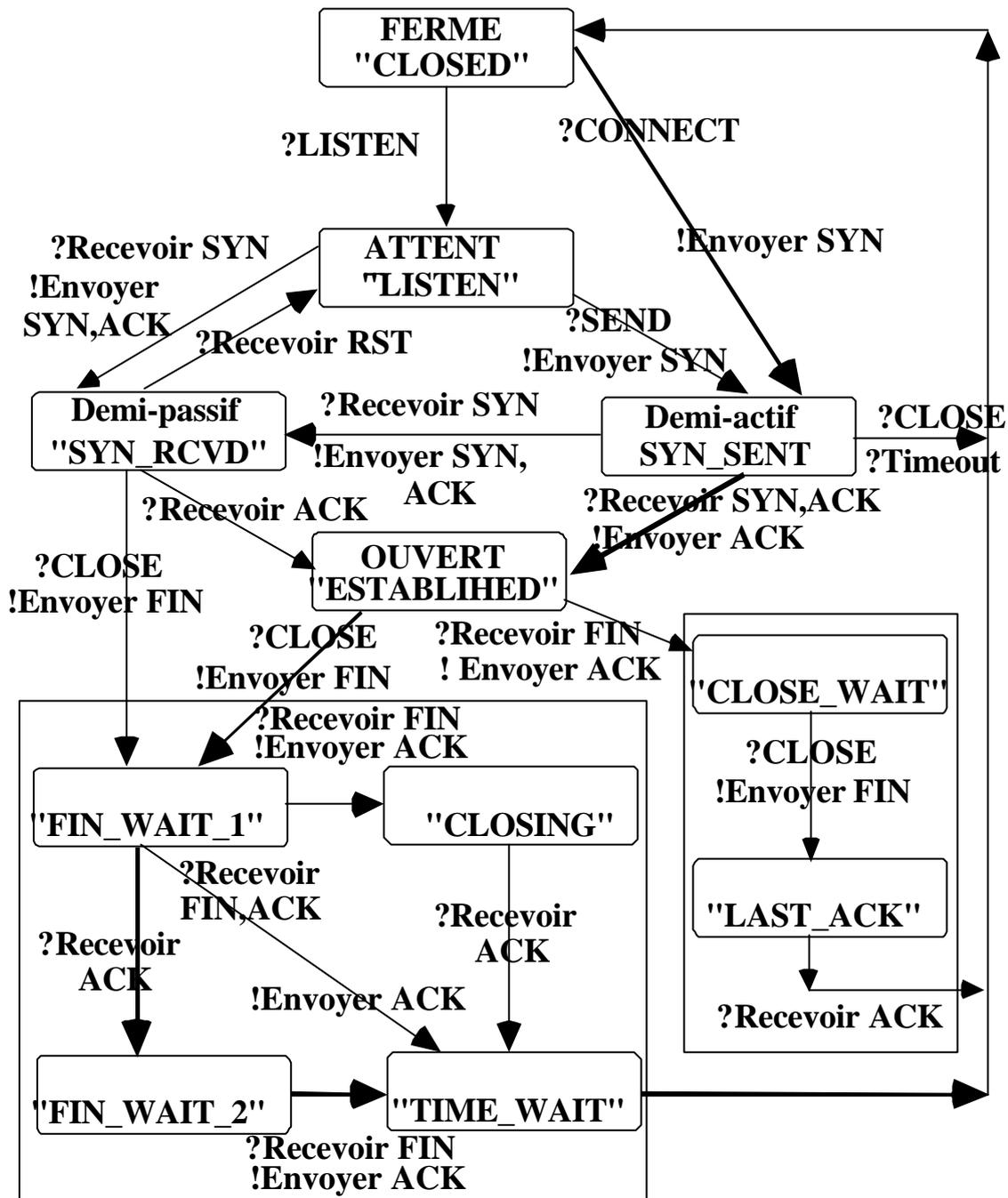
<i>Exercice 13 : DNS ('Domain Name System') Sujet 2.....</i>	<i>84</i>
<i>Exercice 14 : Services et protocoles d'annuaires répartis.....</i>	<i>85</i>
<i>Exercice 15 : Messagerie Internet SMTP.....</i>	<i>91</i>
<i>Exercice 16 : DNS et messagerie Internet/l'approche SPF.....</i>	<i>96</i>
<i>Exercice 17 : Messagerie Internet: Analyse du contenu d'un courrier.....</i>	<i>98</i>
<i>Exercice 18 : Utilisation d'un résolveur DNS.....</i>	<i>101</i>
<i>Exercice 19 : IMAP ('Internet Message Access Protocol').....</i>	<i>104</i>
<i>Exercice 20 : WAP ('Wireless Application Protocol').....</i>	<i>106</i>
<i>Exercice 21 : Services sur la toile ('Web Services').....</i>	<i>111</i>
<i>Exercice 22 : Protocole SIP ('Session Initiation Protocol').....</i>	<i>115</i>
<i>Exercice 23 : Protocole UPnP ('Universal Plug and Play').....</i>	<i>120</i>

<p style="text-align: center;"><b>PREMIER CHAPITRE</b></p> <p style="text-align: center;"><b>EXERCICES CONTROLE REPARTI</b></p>
---

## **Série d'exercices mode message**

### **Exercice 1: Automate de connexion TCP**

- 1) Citez différentes raisons qui justifient l'introduction de la couche transport dans la pile des protocoles de communication OSI.
- 2) Rappelez les principaux choix de conception du protocole de transport TCP.(quelles sont les fonctions réalisées par TCP?).
- 3) Une vision partielle de l'automate de l'ouverture de connexion et de la fermeture de connexion en TCP est donnée par la figure page suivante. Déterminez les éléments de service et les éléments de protocole utilisés dans cet automate?
- 4) Examinez l'automate en suivant les transitions portées en traits gras. Il s'agit d'une ouverture de connexion suivie d'une fermeture. Analysez les principales options de conception de cette séquence de connexion déconnexion en TCP.
- 5) Complétez la lecture de l'automate en examinant d'autres états et d'autres transitions (ouverture passive, ...).



# **Série d'exercices : Appel de procédure distante**

## **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 considère 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 distance à chaque fois.

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

Proposez une organisation pour rendre efficacement un tel service.

## Exercice 6 : 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 ?

# Série d'exercices : Etudes de cas

## Exercice 7 : Conception d'une application de niveau session

Une application de longue durée (de type traitement par lots ou transfert de fichier) est réalisée à distance à partir d'un ordinateur émetteur E sur un ordinateur récepteur R. Le site E émet des suites de messages correspondant à des données à transmettre (par exemple des articles d'un fichier). La conception de l'application doit prendre en compte des objectifs de reprise et amène donc à transmettre les données par groupe de 100 messages et à valider la transmission des messages d'un groupe avant de passer au suivant.

1 Le concepteur utilise la notion de point de synchronisation définie dans la couche session OSI et utilisable par les applications au moyen des protocoles RTS et présentation OSI.

Rappelez la définition et l'usage des différentes notions de synchronisation mineure, majeure, dialogue de session, activité de session.

Le service de synchronisation majeure comporte quatre unités de service qui sont:

- S-SYNC-MAJOR.request
- S-SYNC-MAJOR.indication
- S-SYNC-MAJOR.response
- S-SYNC-MAJOR.confirmation

Donnez l'automate du service de demande de pose de point de synchronisation majeur (c'est l'automate de service du côté émetteur qui demande la pose). Cet automate ne comporte donc que des émissions d'unités de service et des arrivées d'unités de service.

On suppose que l'émetteur possède le jeton de synchronisation majeure et d'activité et qu'il peut donc poser un point de synchronisation majeur. Pour cet automate on ne s'intéresse qu'au mode de fonctionnement normal dans lequel aucune panne n'intervient (ni panne de site, d'entité distante, ...).

Pour chaque état on définira clairement en une phrase la situation à laquelle correspond cet état. Pour chaque transition on donnera la condition et l'action associée en les justifiant.

2 On souhaite ajouter au mécanisme précédent l'acquisition du jeton de synchronisation majeure et d'activité lorsque le site n'en dispose pas (par exemple au début de l'échange).

Les unités de service pour la demande du jeton sont:

- S-TOKEN-PLEASE.request
- S-TOKEN-PLEASE.indication

Les unités de service pour la cession du jeton sont:

- S-TOKEN-GIVE.request
- S-TOKEN-GIVE.indication

On suppose l'existence d'une variable booléenne "jeton\_majeur" indiquant la présence du jeton majeur. Donnez l'automate du service de demande de cession et d'obtention du jeton majeur. Là encore on ne s'intéresse qu'au fonctionnement sans pannes.

3 On souhaite enfin définir le comportement de l'émetteur qui transmet entre deux synchronisations majeures 100 messages de données normales au moyen de:

- S-DATA.request
- S-DATA.indication

Donnez l'automate de ce dernier comportement (dans les mêmes conditions que précédemment).

4 En déduire l'automate complet comprenant vos réponses aux questions 2, 3, 4.

## Exercice 8 : Comparaison de la programmation d'applications réparties avec TCP et avec CORBA

Pour comparer les deux modes de communication 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 et la solution en mode RPC utilise l'approche objets répartis CORBA.

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 d'ordre général concernant les aspects réseaux des deux solutions.

### *A) Utilisation du niveau transport TCP avec l'interface Socket*

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 gethostbyname. 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;
strcpy (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.len;
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) ?

## **B) Utilisation de l'approche objets répartis avec Corba**

On se propose d'étudier la solution au problème de cotation précédent en utilisant l'approche objet réparti CORBA.

7) Rappelez les principes généraux de l'approche CORBA ? Comment est réalisée l'invocation à distance en CORBA ?

Le code IDL suivant décrit l'accès au serveur de cotation :

```
module Stock {
exception Invalid_Stock {};
interface Quoter {
long get_quote (in string stock_name)
raises (Invalid_Stock);
};
};
```

8) A quoi sert la déclaration 'interface Quoter'. A quoi sert la déclaration 'long get\_quote (in string stock\_name)' (à quoi correspond elle dans le programme socket) ? A quoi sert l'exception exception Invalid\_Stock {}; (qu'est ce qu'elle remplace dans le programme socket) ?

Un programme client CORBA pour l'invocation du service de cotation est le suivant:

```
int main (int argc, char *argv[])
{
// Nom du service de cotation invoqué.
const char *name = "Quoter";
Name service_name;
service_name.length(1);
service_name[0].id = name;

// Obtention objet obj invoqué localement (tous les traitements
// nécessaires sont résumés par une procédure bind_service)
Object_var obj = bind_service (argc, argv, service_name);

int result = 1;
try {

// Obtention d'une référence q sur obj (avec vérification de type)
Quoter_var q = Quoter::_narrow (obj);

// Invocation sur obj du service de cotation distante pour la valeur IBM
const char *stock_name = "IBM";
long value = q->get_quote (stock_name);

// Edition du résultat et traitement des erreurs
cout << "value of " << stock_name
<< "=$"
<< value << endl;
result = 0; // Success!
}
}
```

```

// Exception BAD PARAM si erreur de type d'interface
catch (CORBA::BAD_PARAM) {
cerr << "_narrow() failed: "
<< service_name
<< " is not a Quoter!";
} catch (Invalid_Stock &) {
cerr << stock_name
<< " is not a valid stock name!\n";
}
return result;
}

```

9) La requête 'Object\_var obj = bind\_service (argc, argv, service\_name);' permet la création d'un objet local au client 'obj' sur lequel est réalisé l'invocation distante. Que est le rôle de obj (quel traitement réalise l'objet obj), quel est le point de vue adopté relativement aux conversions dans ce programme (quel est le niveau d'interopérabilité de la solution) ?

## Exercice 9 : Objets répartis en CORBA

### Généralités CORBA

- 1) Les protocoles de communication en mode client-serveur réalisés en mode message utilisent au moins deux messages principaux. Quels sont ces messages ? Donnez leur nom dans le contexte de CORBA ?
- 2) Dans les approches à RPC comme celle utilisée dans CORBA on utilise des souches ou talons (ou encore en anglais stubs, skeletons,...). Quels sont les rôles assignés aux souches ?
- 3) Dans ses objectifs, il est indiqué que CORBA permet l'interopérabilité entre différentes plateformes matérielles/logicielles. Indiquez ce qui dans CORBA assure cette interopérabilité. Indication: comment un client en C++ avec un ORB Corba (par exemple le produit Orbacus de la société IONA) sur en Windows peut-il dialoguer avec un serveur en Java avec un ORB Borland Visibroker en Unix.

### Langage IDL

- 4) Le langage IDL CORBA est un langage qui ne possède pas d'instructions de traitement. Pourquoi ?
- 5) Un compilateur C++ génère du code chargeable et exécutable. Qu'est ce que génère un compilateur IDL ?

Le code IDL suivant définit quelques éléments pour une interface d'accès à des fonctions de comptabilité.

```
module CORBACompta
{
  struct Compte {
    attribute string IdentCompte ;
    attribute string TypeCompte ;
    attribute Client client ;
  };
  struct Client {
    attribute string NomClient ;
  };
  typedef sequence<Compte> SeqCompte ;
  typedef sequence<Client> SeqClient ;

  interface GestionFamilledeCompte {
    Compte obtenirCompte (in string IdentCompte) ;
    SeqClient recherchepartypeCompte (in string TypeCompte);
    SeqCompte rechercheparnomclient (in string NomClient);
  };
};
```

6) Commentez cette définition. Quels sont les variables, les types et les opérations prévues à ce niveau d'implantation ?

7) Le code IDL précédent offre des avantages de performances par rapport à une conception dans laquelle les types construits (comme Compte ou Client) seraient définis comme des interfaces. Pourquoi ?

## **Architecture CORBA**

8) CORBA définit dans son architecture l'existence d'un annuaire d'interfaces ('interface repository'). A quoi sert cet annuaire ?

9) CORBA définit dans son architecture un annuaire d'implantations ('implementation repository'). A quoi sert cet annuaire ? Qui l'utilise principalement ?

10) CORBA réalise pour un utilisateur des fonctions habituelles dans un protocole réseau de multiplexage à l'émission et de démultiplexage à la réception. Expliquez en quoi. Comment est assurée cette fonction de multiplexage/démultiplexage par CORBA ?

## **Corba et l'approche objet**

11) CORBA se place dans le cadre des approches objets. L'une des caractéristiques majeure de l'approche objet est l'encapsulation (en anglais 'information hiding'). En quoi CORBA assure-t-il le respect du principe d'encapsulation ?

12) L'un des éléments considérés comme très important dans le développement de l'approche objet a été la possibilité de réutilisation de code par héritage. Comment l'héritage est-il traité dans CORBA ?

# SECOND CHAPITRE

## EXERCICES PRÉSENTATION DES DONNEES

### Série d'exercices conversions

#### Exercice 1 : Optimisation des conversions

Lorsque l'on réalise des échanges sur des systèmes informatiques qui utilisent des formats internes différents, on utilise (par exemple en ASN1) un format de présentation des données commun à toutes les machines du réseau. On doit donc convertir à l'émission les données en ce format réseau et restituer à la réception le format de la machine cible.

Dans le cas où les sites émetteurs ou récepteurs sont de même type ce travail est inutile. Que peut-on faire pour optimiser les traitements?

#### Exercice 2 : Problème des formats de mémoire (petit boutiste, grand boutiste)

On considère dans un ordinateur 16 bits le rangement d'un enregistrement de type chaîne de caractères en mémoire centrale.

Une chaîne est constituée par la donnée :

- de sa longueur sous forme d'un entier sur 16 bits
- de la suite des caractères octet par octet.

*Par exemple* : pour la chaîne de caractères dans laquelle  $\zeta$  matérialise un espace

" êtr $\zeta$ bou $\zeta$ bn $\zeta$ bpas $\zeta$ bêtr $\zeta$ b"

On mémorise :

. un mot qui représente l'entier "0014" en hexadécimal pour dire que la chaîne fait 20 caractères

. la chaîne elle-même " êtr $\zeta$ bou $\zeta$ bn $\zeta$ bpas $\zeta$ bêtr $\zeta$ b".

**Question 1** Le problème réside dans la façon de stocker la représentation d'une donnée, et en particulier d'un entier, dans le mémoire d'une machine. Il existe deux solutions :

- \* petit boutiste ("little endian")
- \* grand boutiste ("big endian").

Rappelez le principe de stockage de ces deux solutions.

**Question 2** On considère deux machines A et B à mots de 16 bits. La machine A utilise la convention petit boutiste et la machine B la convention grand boutiste pour le rangement des octets en mémoire. On suppose, pour A et B, que les bits d'un mot mémoire sont toujours numérotés de

gauche à droite 

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Donnez l'image de la mémoire dans les deux cas octet par octet de l'enregistrement donné plus haut. Les caractères successifs d'une chaîne sont rangés en mémoire dans l'ordre croissant des adresses d'octets pour les deux solutions : premier caractère dans la première case mémoire, deuxième caractère dans la case suivante. On numérote les octets et les mots mémoire à partir de 0.

**Question 3** Dans une première expérience de communication réseau de la machine A vers la machine B on extrait l'information de la machine A octet par octet dans l'ordre où ils sont rangés. On les stocke sur la machine B dans l'ordre d'arrivée.

Quel est le résultat obtenu ? Pourquoi est-il faux?

**Question 4** Dans une seconde expérience de communication on inverse à l'arrivée chaque couple d'octet : le premier octet du couple est permuté avec le deuxième.

Quel est le résultat obtenu ? Pourquoi est-il faux ? Quel serait le texte affiché ?

**Question 5** Quel est le remède aux deux problèmes précédents adopté dans ASN1 ?

### Exercice 3 : 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'abréviation 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-identif er ::= **INTEGER**

Transfer-syntax-name ::= **OBJECT IDENTIFIER**

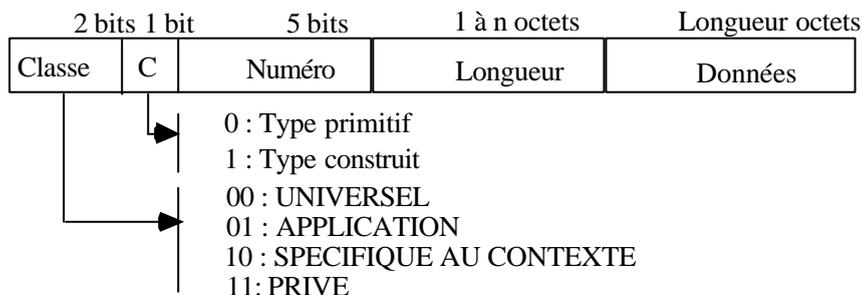
- 1 Le concepteur a défini deux formats principaux de données utilisateurs. Quels sont-ils? En fonction de leurs définitions a quels besoins correspondent t'ils selon vous?
- 2 La définition de la syntaxe de transfert utilisée est optionnelle. Pourquoi ce choix est-il effectué?
- 3 La définition du contexte 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 4 : 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 5 : 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 6 : Problème général ASN1**

On souhaite construire un serveur d'annuaire téléphonique simple. Ce serveur d'annuaire peut être consulté et mis à 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.

## Exercice 7 : Définition d'une traduction du langage IDL de CORBA vers le langage Pascal

Vous disposez d'une application comprenant plusieurs dizaines de milliers de lignes écrites en Pascal (Cf. en fin de sujet la note sur le langage Pascal). Vous avez été chargé d'intégrer cette application dans un environnement CORBA. On vous demande donc de réaliser une étude pour interfacier cette application avec des programmes CORBA existants : on souhaite que ces programmes puissent invoquer l'application écrite en Pascal et puisse lui soumettre des requêtes.

Cette étude comprend deux parties :

- la définition d'une traduction du langage IDL de CORBA vers le langage Pascal
- un certain nombre de modifications des sources Pascal de l'application

Nous ne nous intéressons ici qu'à la première partie de l'étude (i.e. la traduction du langage IDL vers le langage Pascal).

### 1/ Langage IDL

#### 1.1/ Barrer la mauvaise réponse.

Le langage IDL de CORBA permet de définir :

- les interfaces des objets serveur d'une application : oui - non
- les implantations des objets serveur d'une application : oui - non

#### 1.2/ Commenter le code IDL suivant en précisant le rôle de chaque mot-clé.

```
module Garage
{
    typedef long typeChevFisc;
    interface Voiture
    {
        attribute long NbPlaces;
        long Vignette( in typeChevFisc cf );
    };
};
```

### 2/ Traduction IDL vers Pascal

2.1/ En vous inspirant des traductions IDL vers respectivement C, C++ et Java du polycopié, proposer une traduction du code IDL de la question 1.2/ vers le langage Pascal.

2.2/ Expliquer en quelques mots, pour chaque mot-clé traduit, vos choix.

### 3/ Passage de paramètres

3.1/ Rappeler le rôle des mots-clés in, inout et out dans la déclaration IDL d'une méthode.

3.2/ Proposer une traduction en Pascal du code IDL suivant :

```
long maMethode( in long arg1 , inout long arg2 , out long arg3 );
```

#### 4/ Traduction des types de données du langage IDL

4.1/ En plus du type IDL long vu aux questions 1/ et 3.2/, citer les autres types du langage IDL pour lesquels il est nécessaire de définir une traduction vers les langages Pascal. En proposer une traduction.

#### Note à propos du langage Pascal

Pour les besoins de ce problème, on considère que le langage Pascal retenu est identique au langage Pascal habituel, mais qu'il comporte un type spécifique OBJECT permettant de définir des classes d'objets.

Par exemple, le code suivant :

```
TYPE tHeure = RECORD (* ... *) END;
TYPE tHorloge = OBJECT
    FuseauHoraire : Integer;
    METHOD RetourneHeure : tHeure;
    METHOD PositionneFuseau( fh : Integer );
END;
```

définit en plus du type tHeure, l'interface d'une classe tHorloge qui comporte : une variable entière FuseauHoraire et deux méthodes RetourneHeure et PositionneFuseau. La définition du code de ces méthodes est identique à celle des procédures Pascal traditionnelles.

## Exercice 8 : CORBA

Pour permettre l'interopérabilité dans des systèmes d'objets répartis de constructeurs différents, le standard CORBA (version 2.0) décrit deux éléments essentiels qui sont le langage de définition des interfaces (IDL "Interface Definition Language") et les protocoles GIOP (General Inter-ORB Protocol) et IIOP (Internet Inter-ORB Protocol).

### 1 Questions IDL CORBA

L'IDL CORBA permet de décrire pour des objets clients, les interfaces d'accès à des objets serveurs (c'est-à-dire les services fournis par les objets serveurs). Nous donnons en annexe la grammaire fort simplifiée du langage IDL que vous pouvez consulter si nécessaire. Pour répondre aux questions qui suivent la compréhension de la totalité de la grammaire IDL CORBA n'est pas nécessaire.

Soit le fichier IDL contenant la définition suivante pour des services fournis par un objet Calcul. Celui-ci opère sur un nombre réel stocké en mémoire :

```
module Math
{
  interface Calcul
  {
    exception DivByZero { String commentaire;} ;
    oneway void Init ( in Double valeur ) ;
    Double Add ( in Double valeur ) ;
    Double Sub ( in Double valeur ) ;
    Double Mult ( in Double valeur ) ;
    Double Div ( in Double valeur ) raises ( DivByZero ) ;
  };
};
```

1.1 Quel est l'ensemble des services ou opérations offerts par le serveur ou l'objet Calcul?

1.2 Que signifie la suite de caractères : '( in Double valeur )' dans l'exemple précédent ?

1.3 En IDL CORBA le mot clé "oneway" définit une invocation de procédure distante asynchrone sans réponse Quel est le mode de fonctionnement d'un tel appel?

1.4 Par défaut, lorsque le mot "oneway" n'est pas précisé, les opérations décrites en IDL sont des appels de procédure distante synchrone avec la sémantique "au plus une fois" vers l'objet ou le serveur distant. Rappelez le fonctionnement d'un appel de procédure synchrone avec une sémantique au plus une fois. Qu'est-ce que cela signifie pour le développeur d'applications réparties?

1.5 Par rapport au langage de syntaxe abstraite ASN.1 étudié en cours et en travaux dirigés, quel est l'apport du langage IDL CORBA précédent?

### 2 Questions GIOP et IIOP

Le bus logiciel CORBA, appelé ORB pour Object Request Broker, sert de support de communication entre les clients et les serveurs. Il véhicule les requêtes et les résultats ou les erreurs associées. Pour permettre l'interopérabilité entre des bus de différents constructeurs, un protocole général GIOP a été défini ("Général Inter-ORB Protocol"). Le protocole d'interopérabilité inter-bus sur l'Internet ou Internet Inter-ORB Protocol (IIOP) est la mise en œuvre du protocole GIOP pour l'architecture TCP/IP. On peut dériver du protocole GIOP d'autres protocoles inter-ORB associés à d'autres architectures de réseau.

Entre autres fonctionnalités le protocole GIOP définit une syntaxe de transfert. La représentation des données au format CDR ("Common Data Representation") utilisée par le protocole GIOP spécifie un format de données pour représenter tous les types IDL sous la forme d'une suite d'octets. CDR décrit précisément le codage et l'alignement de chacun des types dans un message.

2.1 Quelle est la solution adoptée par ASN1/BER relativement au problème de l'ordre d'émission des octets (petit boutiste, grand boutiste)?

2.2 L'un des premiers octets du message GIOP contient la valeur d'un booléen indiquant l'ordre des octets des données encapsulées. Si le booléen est à faux, les octets sont alignés dans l'ordre grand boutiste ("big endian"); si la valeur est à vrai, les données sont encodées dans l'ordre petit boutiste ("little endian"). Donner les raisons du codage de ce premier booléen dans un message.

Pour pouvoir accéder aux services offerts par un serveur ou un objet, il faut être capable de l'identifier et de le localiser et de réaliser des RPC. C'est l'une des fonctions du bus logiciel que de gérer des références d'objets à cette fin. Une référence d'objet est baptisée (IOR Interoperable Object Reference). Il n'y a pas un format unique de référence car les développeurs sont libres de cette définition.. Par exemple ces références contiennent des informations spécifiques à la couche de transport utilisée pour accéder à l'objet. Cependant toute référence doit au moins contenir les informations suivantes:

- le numéro de version de la spécification de la couche transport acceptée par le serveur d'objet.
- l'adresse transport permettant d'atteindre le serveur destinataire de la requête (dans le format du protocole de transport utilisé).
- une clé d'accès représentée par une séquence d'octets permettant d'identifier et de localiser l'objet sur le serveur.

Un exemple de référence d'objet IOR qui est fourni par l'ORB industriel ILU est donné par:

```
1.0:sunrpc_2_0x61a79_153029598/rm=tcp_163.173.128.208_3503:IDL%3AMath%3A/Math/Calcul
```

### **2.3 Pour cet exemple donner la signification des différents champs.**

2.4 Dans l'exemple de l'IOR précédent, quels sont les protocoles de transport et de RPC utilisés par l'ORB pour réaliser les invocations distantes?

2.5 Le protocole IIOP ne fonctionne pas en mode connecté. Par contre il utilise pour envoyer ses messages une voie de communication TCP qui elle est en mode connecté. Citez au moins deux protocoles étudiés en cours fonctionnant également en mode non connecté au dessus de TCP?

Le protocole GIOP définit 7 types de messages différents qui sont émis soit par le client, soit par le serveur, soit par les deux.

**Request** : émis par le client pour invoquer des opérations sur les objets ou serveurs CORBA. L'entête du message contient entre autres un identificateur de requête. Le corps d'un message request contient les valeurs des paramètres en entrée (mode in et inout du langage IDL).

**Reply** : émis par le serveur en réponse à un message **Request**. L'entête du message contient l'identificateur de la requête. Le corps du message contient le résultat de l'opération ainsi que les paramètres en sortie de celle-ci (mode inout et out). Si un problème survient lors du transport ou de l'exécution de la requête, le message contient alors la valeur de l'exception indiquant ce problème.

**CancelRequest** : permet à un client d'informer un serveur qu'il ne désire plus attendre la réponse à une requête qu'il a émis.

**LocateRequest** : ce message permet de savoir si une référence IOR est valide, si le serveur est capable de recevoir directement des requêtes pour un objet donné, et dans le cas contraire, de déterminer la nouvelle adresse IOR permettant d'accéder à l'objet. Cela permet de gérer la migration des objets.

**LocateReply** : c'est la réponse associée à un message de type LocateRequest. Il contient un booléen indiquant si la référence IOR désigne un objet local. Si ce booléen est à faux, alors le message contient en plus une référence IOR indiquant la nouvelle localisation de l'objet.

**CloseConnection** : ce message informe le client que le serveur interrompt son service. Les clients savent alors que toutes les requêtes pour lesquelles ils attendaient des réponses ne seront jamais traitées. Les clients peuvent retransmettre leurs requêtes vers d'autres serveurs.

**MessageError** : ce message est envoyé en réponse à tout message GIOP qui ne peut pas être traité car il est erroné. Par exemple, le numéro de version est inconnu, un type de message est inconnu, ou un en-tête est erroné.

L'en-tête d'une requête (**Request**) se définit principalement en IDL de la façon suivante:

```
struct RequestHeader {
    unsigned long          request_id;
    boolean                reponse_expected;
    sequence<octet>       object_key;
    string                 operation;
};
```

2.6 Donner la signification de chaque champ et leur fonction.

2.7 Dans le cas d'une opération définie en IDL de la façon suivante :

Double exemple (in Short m, out String str, inout Double p);

Quels sont les éléments qui vont faire partie du corps de la requête ?

2.8 On s'intéresse dans cette question à la réalisation du protocole IIOP au dessus de la couche transport TCP (interface socket). En supposant qu'il n'y ait encore eu aucune requête entre le client et le serveur, un client décide de demander la réalisation d'une requête à un serveur. Une connexion TCP doit être établie puis le message de requête doit être acheminé. On utilise l'interface socket pour TCP.

Donner l'enchaînement des primitives socket qui sont nécessaires à la réalisation d'une requête sur un objet distant. Vous commenterez votre solution en donnant la signification des différentes étapes successives proposées aussi bien pour la souche côté client que pour la souche côté serveur

### Annexe : Grammaire de l'IDL CORBA

Nous rappelons quelques conventions d'écriture d'une grammaire d'un langage: les mots en **gras** sont des noms terminaux ou mots-clés, les mots entre  $\langle \rangle$  sont des mots qui sont ensuite redéfinis, les mots entre [ ] sont des mots optionnels. Lorsqu'une déclaration ou un mot est suivi d'une \*, cela signifie que on peut trouver de 0 à n fois cette déclaration ou ce mot. Le texte entre /\* et \*/ est un commentaire.

Un module est un espace de nommage permettant de décrire les interfaces de service offerts par un serveur ou objet distant. Dans une interface on trouve la déclaration de types, de constantes, d'attributs, d'exceptions et d'opérations.

```
module <identificateur> /* un contexte */
{

interface <identificateur> /*interface d'un service ou d'un
objet*/
{
    <declaration de type>*;
    <declaration de constante>*;
    <declaration d'exception>*;
    <declaration d'attribut>*;
    <declaration d'operation>*;
};
};
```

<declaration de type> ::= **typedef** <type> <ident\_type>

<declaration de constante> ::= **const** <identificateur> '='  
<expression>

<declaration d'exception> ::= **exception** <ident\_except>  
'{<parametres>\*'}

```

<declaration d'attribut> ::= <mode att> attribute <type>
<identificateur>

<mode att> ::= [readonly]

<declaration d'operation> ::= <mode d'invocation> <type
retour> <identificateur> '(' <parametres>*)' <clause
d'exception>

<mode invocation> ::= [oneway]

<type retour> ::= void | <type_IDL>
/*Le mot clé void définit l'absence de paramètre résultat*/

<parametres> ::= <mode param> <type> <identificateur>

<mode param> ::= in | out | inout

<clause exception> ::= [ raises '(' <liste_d_exception>')' ]

<type> ::= Short | Long | LongLong | Float | Double | Char
| String | Boolean | ident_type

<liste_d_exception> ::= <ident_except>*

<ident_except> ::= <identificateur>

<ident_type> ::= <identificateur>

<identificateur> ::= <lettre>{<caractere>}*

<caractere> ::= <lettre> | <chiffre> | '_'

<lettre> ::= 'a' | ... | 'z' | 'A' | ... | 'Z'

<chiffre> ::= '0' | ... | '9'

```

Les mots-clés **Short**, **Long**, **LongLong**, **Float**, **Double**, **Char**, **String**, **Boolean**, correspondent à des définitions de types primitifs et signifient respectivement entier court sur 16 bits, entiers long sur 32 bits, entier long sur 64 bits, nombre réel sur 32 bits, nombre réel sur 64 bits, caractère 8 bits ISO latin-1, chaîne de caractères et boolean.

# Série d'exercices sécurité

## Exercice 9 : Cryptogramme 1

1) Cassez le cryptogramme suivant qui utilise une substitution monoalphabétique. Le texte en clair ne contient que des lettres. C'est un extrait d'une très célèbre fable de La Fontaine.

gcxobwryv ib ogx tb syiib  
ypsyxg ib ogx tbv qmgzev  
t cpb wgqrp wrox qysyib  
g tbv obiybwv t roxrigrpv  
vco cp xgeyv tb xcojcyb  
ib qrcsbox vb xorcsz yzv  
ab igyv vb g ebpvbo ig syb  
jcb wyobpx qbv t bcd gzyv  
ib obfgi wcx wrox mrppbxb  
oybp pb zgpjjcgyx gc wbvxyv

2) En supposant que vous disposiez d'un histogramme de la répartition des lettres et des digrammes dans la langue française ainsi qu'une fonction Nombre\_fautes(texte) qui donne (selon un correcteur orthographique) le pourcentage de mots dans un texte qui ont au moins une faute d'orthographe, imaginez un algorithme de cryptanalyse de ce type de crypto-systèmes

## Exercice 10 : Cryptogramme n° 2

Casser le cryptogramme suivant qui utilise une transposition par colonnes. Le texte en clair est issu d'un ouvrage classique sur l'informatique; on peut donc supposer que le mot "ordinateur" y figure. Le texte ne contient que des lettres (sans espace). Il est découpé en blocs de 5 caractères pour plus de lisibilité.

ntsus ueire eibps etsio ootuu rpmm eaicq iunps cnlog  
euern lndur raose xnntu dnaeo eseue clton nretd trels

## Exercice 11 : Cryptogramme n° 3

1 Chiffrer avec le chiffre de Vigenère le texte suivant "textesecretadecoder" en utilisant comme clef le mot crypto

2 Pour le même texte en clair on obtient le texte chiffré suivant "brqksmzcspxiqxtcxzr". Quel est la clef?

3 Supposons que vous disposiez d'un texte en clair et une partie du même texte chiffré, mais que ce texte soit plus court que la clef (par exemple vous ne connaissez que brq dans l'exemple précédent). Quelle information cela vous apporte t'il ? Imaginez des stratégies de cryptanalyse dans le cas ou la

clef est un mot français et dans le cas où c'est une suite aléatoire de lettres. Comment distinguer à priori ces deux cas ?

## Exercice 12 : Etude du système de chiffrement à clé publique RSA

On rappelle l'algorithme du MIT ( Rivest, Shamir, Adleman, 1978).

1- Choisir deux nombres  $p$  et  $q$  premiers et grands ( $p, q > 10^{100}$ )

2 - Calculer  $n = p * q$      $z = (p - 1) * (q - 1)$

3 - Soit  $d$  un nombre premier avec  $z$  ( $z$  et  $d$  sont premiers entre eux).

4 - Déterminer un entier  $e$  tel que  $e * d = 1 \pmod{z}$ .

Soit  $A$  un message binaire. Découper  $A$  en blocs de  $k$  bits tel que  $k$  soit le plus grand entier tel que  $2^k < n$ .

Soit  $P$  un bloc de  $k$  bits, le codage de  $P$  est donné par :  $C = E(P) = P^e \pmod{n}$

Le déchiffage d'un message crypté  $C$  est donné par:  $P = D(C) = C^d \pmod{n}$

1)- On donne les valeurs numériques suivantes :  $p = 3$ ,  $q = 11$  (trop petites en pratique, mais traitable en exercice).

Calculer les valeurs des nombres  $d$  et  $e$  vérifiant les conditions de l'algorithme du MIT. Pour avoir un couple unique on prend la plus petite valeur possible de  $d$  et pour cette valeur la plus petite valeur possible de  $e$ .

Quelle est la clé publique et quelle est la clé secrète?

2) - Soit le message de 3 chiffres 1, 6, 15 soit par blocs de 5 bits la configuration de bits suivante:

00001 00110 01111

Coder ce message en utilisant les paramètres de chiffrement RSA précédents.

3) - On reçoit le message suivant par blocs de 6bits (4 , 14 , 24):

000100 001110 011000

Donner la valeur initiale du message (texte en clair), en prenant les mêmes valeurs pour  $d$ ,  $e$  et  $k$  qu'à la question 2.

Question 4 - Pourquoi ne peut-on prendre  $p$  et  $q$  petits ?

Que se passe-t-il lorsque  $p$  et  $q$  sont de l'ordre de  $10^{10}$  ?

## Exercice 13 : Kerberos

Le projet Athena du MIT (Massachusetts Institutes of Technology) était, dans le milieu des années 80, un projet de recherche portant sur les systèmes répartis. Il avait pour objectif la définition d'un environnement homogène d'accès pour des PC sous MS/DOS, des stations de travail ou des calculateurs sous UNIX en grand nombre et reliés par plusieurs réseaux locaux. L'un des points forts de cet environnement est le protocole qui gère à la fois l'authentification des utilisateurs et certains mécanismes de protection. Ce protocole est dénommé Kerberos (Cerbère, le chien à trois têtes, gardien de l'enfer). Depuis ces origines Kerberos a connu de nombreuses variantes et a été implantés dans différents produits (En particulier chez Digital Equipment et plus récemment chez Microsoft). Il constitue le standard des systèmes de distribution des clefs symétriques (gardiens des clefs) et est très utilisé dans l'Internet, en particulier par les organismes de recherche et d'enseignement. La plus part des protocoles de l'Internet (PPP, HTTP, SMTP, FTP...) supportent une authentification des entités communicantes basées sur Kerberos. Ses principes ont été repris dans des architecture de protection comme DCE et Windows NT (c. f. le chapitre sur la protection de cette seconde partie). La version en cours de normalisation à l'IETF est la v5 qui n'utilise que la cryptographie symétrique (DES et triple DES) et les fonctions de hachage à sens unique (MD5 et SHA-1), la v4 faisant l'objet de la RFC1501.

### Questions 1

Quelle sont les fonctions que doit réaliser un gardien de clefs ?

Expliquer les principes du triple DES et ses avantages par rapport au DES ?

A quoi peut servir la combinaison d'une fonction de hachage et d'un crypto système symétrique ?

### Description du protocole

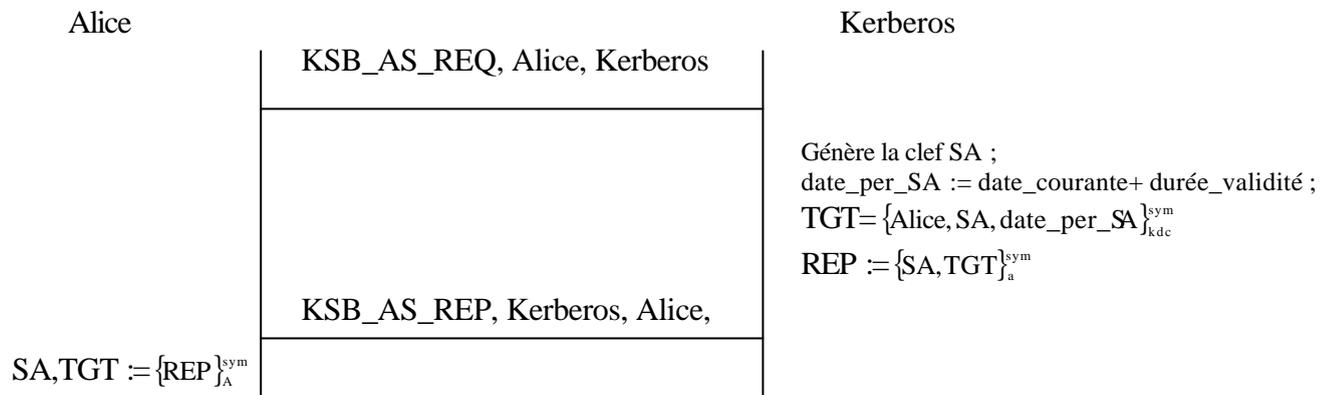
Kerberos met en communication trois entités :

- l'entité client : un programme ou un utilisateur sur une machine donnée. Pour simplifier nous considérons que l'entité client figure uniquement un utilisateur humain. Pour nous ce sera Alice.
- l'entité serveur: un programme requis par l'entité client (par exemple le serveur de fichier). Ce sera dans la suite le serveur de Bob.
- le centre de distribution des clefs, appelé KDC (pour Key Distribution Center), il connaît les clefs privées de toutes les entités clients ou serveurs qu'il conserve dans une base de données, il génère aussi des clefs de session (temporaires).

Pour fonctionner Kerberos nécessite une synchronisation des horloges de toutes les machines du réseau. Ce protocole doit réaliser entre tout couple d'horloge du réseau un écart inférieur à 5 mn pour des dates lues au même instant (ce qui est assez facile à réaliser).

Pour pouvoir utiliser un serveur, Kerberos réalise un protocole de distribution de clés de session qui respecte les étapes suivantes :

1. Alice s'authentifie auprès du KDC
2. Alice demande un ticket d'utilisation du serveur Bob au KDC. Un ticket au sens de Kerberos comporte essentiellement une clé de session et différentes informations annexes détaillées plus loin. KDC lui renvoie un ticket valable pour le serveur demandé.
3. Quand Alice sollicite le serveur Bob, elle utilise le ticket que lui a remis le KDC qu'il met dans sa requête.



Nous allons détailler les trois étapes précédentes. Pour cela, nous définissons précisément les notations des différents objets manipulés par le protocole :

Alice	Le nom du client.
A,a	Une clef utilisée par Alice pour s'authentifier qui en général est dérivée d'un mot de passe A est connue d'Alice et de Kerberos (en majuscule quand elle est utilisée pour le déchiffrement, en minuscule pour le chiffrement)
SA,sa	La clef de session d'Alice SA est connue d'Alice et de Kerberos
Bob	Le nom du serveur invoqué.
B,b	La clef privée du serveur Bob. (en majuscule quand elle est utilisée pour le déchiffrement, en minuscule pour le chiffrement)
Kerberos	Le nom du serveur de distribution des clefs
KDC,kdc	La clef de Kerberos connue de lui seul(en majuscule quand elle est utilisée pour le déchiffrement, en minuscule pour le chiffrement)
KAB,kab	La clef de session déterminée entre client et serveur. (en majuscule quand elle est utilisée pour le déchiffrement, en minuscule pour le chiffrement)
Ticket	Le ticket donné à un client Alice pour utiliser un serveur Bob.
date_fab_X	La date de fabrication de l'objet X
date_courante	Date donnée par l'horloge locale de la machine

Comme dans tout système d'accès a mots de passe, le client a convenu préalablement avec l'administrateur système d'un nom (Alice), et d'un mot de passe qui devient la clef privée A d'Alice. Pareillement, il a été convenu, pour le programme serveur, d'un nom (Bob), et d'une clef privée B. Les clefs ne sont connues que de leur propriétaire (qui doivent donc les garder secrètes), et du serveur d'authentification Kerberos. Ce système doit donc être particulièrement bien protégé.

#### Etape 1 : Authentification du client

Alice commence par envoyer une requête d'authentification KSB\_AS\_REQ contenant son Nom de login (Alice) à Kerberos. Kerberos lui renvoie un message KSB\_AS\_REP contenant une clef de session SA personnelle à Alice et un ticket de contrôle d'accès TGT (Ticket Granting Ticket).

### Questions 2

Expliquez l'usage des deux chiffrements réalisés dans cette étape.

Contre quel type d'attaque est destinée la variable  $date\_per\_SA$  ? Expliquez son usage

Etape 2 : Obtention du ticket d'accès au serveur

Lorsque Alice désire (durant la période de validité de son ticket d'authentification) utiliser le service du serveur Bob, elle adresse à Kerberos une requête d'accès à ce serveur. Cette requête contient outre le nom du serveur, TGT et la date courante chiffrée avec SA.

A réception de cette demande Kerberos déchiffre TGT, récupère la clef SA, déchiffre la date avec SA et vérifie que cette date est voisine de son heure courante.

### Questions 3

Que prouve ce contrôle ?

Quelle technique de protection peut utiliser Kerberos pour le contrôle des droits d'Alice?

Quel est l'usage du chiffrement réalisé par Kerberos?

Etape 3 : Accès au serveur applicatif

Dans un délai compatible avec la durée de validité du Ticket, Alice va demander au serveur Bob d'exécuter la transaction pour laquelle elle a obtenu le ticket. Elle envoie donc à ce serveur une requête avec le ticket.

### Questions 4

A quoi sert la variable d1 et pourquoi est elle chiffrée ?

Expliquez le principe du contrôle réalisé par Bob

Quel est l'usage de la variable REP2 ?

A quoi pourra servir KAB dans la suite des échanges de la transaction courante entre Alice et Bob ?

Comment appelle t'on une telle variable?

## Extension de Kerberos

Une extension de Kerberos est en cours de normalisation. Elle consiste à utiliser un crypto système asymétrique pour toutes les opérations utilisant des clefs rémanente (A,B, KDC). Dans ce cas Kerberos n'a plus besoins de stocker les clefs symétriques de chaque participant. Il ne stocke que des clefs symétriques de session et sa clef (KDC) et des certificats pour chaque utilisateur. Ceci réduit considérablement le risque lié à une attaque de Kerberos.

### Question 5

Donnez le schéma d'échange de messages de l'étape 1 (cas nominal) correspondant à cette extension en justifiant l'utilisation de chaque opération cryptographique.

[**Adr\_c ; Adr\_s ; ( { Ac }Clé\_sess\_c,s ; { Tc,s }Clé\_s )** ]

## Exercice 14 : Partage d'un secret

Un conseil d'administration prend ses décisions au moyen de votes électroniques. Il comporte un président dont la voix compte double et trois membres dont la voix est simple (notés 1,2,3).

Une décision n'est prise que si un vote rassemble au moins 3 voix.

Les votes sont acquis lorsque le titulaire présente selon son rang soit une soit deux parts d'un secret selon ses droits qui l'authentifie.

- Les services de sécurité ont fait implanter la méthode de Shamir avec une arithmétique modulo 31.
- Le secret partagé est tiré aléatoirement à la valeur 7
- Les coefficients aléatoires du polynôme sont 1, 9, et le terme constant 7
- On déduit les parts de secret du président aux points  
1, 2
- On déduit des parts du secret des membres 1, 2, 3 des valeurs du polynôme aux points  
3, 4, 5

1 Quelles sont les valeurs des parts?

2 On suppose que le président et le membre 1 sont d'accords.  
Montrez qu'ils peuvent emporter le vote.

## Exercice 15 : Problème de notarisation

On souhaite réaliser un service ayant pour objectif la non répudiation de transactions entre deux sites A et B qui fonctionnent en mode A client et B serveur (service de notarisation). Le notaire est noté N.

On note  $a$  la clef de chiffrement (clef publique dans les crypto systèmes asymétriques) de Bob et  $b$  sa clef de déchiffrement (clef privée dans les crypto systèmes asymétriques).

Les étapes du protocole sont les suivantes :

1. Alice chiffre et signe  $M$  avec sa clef  $a$  et l'envoie au notaire.
2. Le notaire déchiffre le message, vérifie la signature qui garantit que le demandeur du service est bien Alice. Il attribue à l'envoi du message un numéro de séquence  $envoi(M)$ , date cet envoi  $te(M)$ , enregistre  $envoi(M)$ ,  $te(M)$ ,  $M$  dans un fichier intègre. Il constitue un message contenant  $envoi(M)$  signé avec la clef d'Alice et  $(envoi(M), te(M), M)$  chiffré avec sa clef.
3. Alice peut alors contrôler la signature qui authentifie le notaire (qui est le seul avec Alice à connaître la clef d'Alice) et constitue une référence de la preuve d'envoi. Par contre Alice ne peut pas déchiffrer la seconde partie du message. Elle l'envoie à Bob.
4. Bob ne peut pas, lui non plus, lire le message. Il le renvoie au notaire
5. Le notaire déchiffre le message avec sa clef. Il vérifie que le numéro d'envoi est dans son fichier. Il attribue à la réception du message un numéro de séquence  $réception(M)$ , date cette réception  $tr(M)$ , enregistre  $réception(M)$ ,  $tr(M)$ ,  $M$  dans le fichier intègre. Il constitue un message contenant  $(envoi(M), te(M), réception(M), tr(m), M)$  signé et chiffré avec la clef de Bob. Il transmet ce message à Bob. Il envoi un message contenant  $réception(M)$  à Alice.

6. Bob peut déchiffrer le message, contrôler la signature qui authentifie le notaire. Il envoie un accusé de réception signé avec sa clé au notaire.
7. Le notaire enregistre la fin de la transaction.

**Question 1** Rappelez la définition du problème de non répudiation.

**Question 2** Décrivez par des diagrammes d'échange de messages chaque étape en utilisant uniquement des algorithmes de chiffrement symétriques; pour chaque étape, exprimez les propriétés obtenues: confidentialité, intégrité, non répudiation, vis à vis de tous les participants (A, B, N, les autres entités présentes dans le réseau) qui sont réalisées par chacune des six étapes.

**Question 3** Proposez pour chacune des étapes une solution à clé publique qui atteigne exactement les propriétés précédentes vis à vis des mêmes intervenants. Expliquez de manière très précise pourquoi votre solution assure les propriétés recherchées.

**Question 4** On suppose que l'utilisateur client A dispose d'un délai  $d_{max}$  pour signaler la perte de sa clé. Toute perte non signalée dans le délai  $d_{max}$  ne peut-être tenue à charge contre l'utilisateur. Toute perte non signalée après le délai  $d_{max}$  entraîne la responsabilité totale du client en cas de contestation.

Que pourrait faire le notaire pour que le serveur B et le notaire ne soient jamais piégés par un client malveillant qui déclare la perte de sa clé après avoir donné un ordre et avant  $d_{max}$ . Dans quelles applications une telle stratégie est-elle utilisable? Dans quelles applications est-elle inutilisable?

**Question 5** La solution utilise l'algorithme RSA. Si un utilisateur pour répudier sa signature électronique conteste la sécurité du protocole quels arguments peuvent être opposés?

**Question 6** Le site B peut essayer de dénier avoir reçu l'ordre (s'il ne lui convient pas) en n'émettant pas l'acquittement final et en prétendant ensuite qu'il n'a jamais reçu la transaction interprétable ou que son ordinateur est tombé en panne à ce moment. Comment le notaire peut-il contraindre cette attitude (recherchez une solution déduite des pratiques bancaires courantes)?

## Exercice 16 : Gestion d'une connexion sécurisée

Dans ce problème sont étudiés les mécanismes de gestion d'une connexion sécurisée, c'est à dire offrant des services de confidentialité ou d'intégrité des données.

On considère que le protocole de sécurisation est construit en utilisant les services d'un protocole de transfert fiable comme le transport ISO TP4, TCP...

### I.1 Principes généraux

Une communication sécurisée nécessite les opérations principales suivantes, elles mêmes divisées en étapes:

#### 1) Ouverture de la connexion sécurisée:

- a) Ouverture de la connexion du protocole sous-jacent,
- b) Authentification (réciproque) des entités communicantes,
- c) Négociation du mode de sécurisation des transferts,
- d) Échange d'une clef de session.

#### 2) Transfert des données:

- e) En émission: chiffrement des messages, construction des trames sécurisées,
- f) En réception: Déchiffrement des messages et détection des attaques éventuelles,
- g) Périodiquement: changement de la clef de session.

#### 3) Fermeture de la connexion sécurisée:

- h) Fermeture de la connexion du protocole sous-jacent,
- i) Destruction du contexte de connexion sécurisée.

Expliquez en deux à trois lignes l'utilité de chacun des 9 points précédents.

Dans la suite de ce problème nous utilisons les notations suivantes:

- A,B... les entités impliquées dans l'échange sécurisé
- A----->B: M (dans le protocole A doit envoyer M à l'entité B)
- S est un serveur de clef
- K est une clef et  $K^{-1}$  son inverse (dans le chiffrement asymétrique K est la clef publique et  $K^{-1}$  la clef privée)
- $\{X\}_K$  est un texte X chiffré avec K comme clef

On a donc  $\{\{\{X\}_K\}_{K^{-1}}\} = X$  et l'on suppose aussi que  $\{\{\{X\}_{K^{-1}}\}_K\} = X$

- On utilise dans ce problème la notion de fonction de hachage H. H(M) application de H à un message M fournit une valeur dépendant de l'ensemble du message M (ce pourrait être une somme de contrôle ou le résultat d'une division polynomiale).

### I.2 Authentification des correspondants

Un schéma d'authentification à clef publique dans la communication sécurisée est le suivant. On utilise un chiffrement asymétrique tel que le RSA.  $K_b$  est la clef publique de B.

Pour que B s'authentifie auprès de A on procède aux opérations suivantes:

- 1) A tire un nombre aléatoire N
- 2) Message 1 A----->B:A,B,N

3) Message 2  $B \rightarrow A: B, A, \{N\}_{K_{b-1}}$

A vérifie que  $\{\{N\}_{K_{b-1}}\}_{K_b} = N$  ce qui authentifie B.

I.2.1 Rappelez les principes du chiffrement asymétrique (à clef publique)

I.2.2 Pourquoi N doit-il être un nombre aléatoire (pas une constante par exemple) ?

I.3 Utilisation d'un serveur de clefs publiques

Pour obtenir la clef publique  $K_b$  de B, A fait appel à un serveur S de clefs publiques. Ce serveur gère un fichier qui ne peut être modifié que par un responsable de sécurité dont la clef publique  $K_s$  est connue de tous les utilisateurs habilités à utiliser le protocole de sécurité.

Un enregistrement du fichier a la structure suivante:

$B, K_b, \{H(B, K_b)\}_{K_{s-1}}$

I.3.1 Quel est l'utilité du dernier champ de cet enregistrement?

I.3.2 Complétez le protocole d'authentification donné à la question précédente en introduisant les opérations entre A et le serveur S pour récupérer  $K_b$  et vérifier sa validité.

I.4 Négociation des paramètres et échange la clef de session

Dans l'échange suivant de négociation et d'échange:

A est l'appelant et B l'appelé.

Y est une proposition du mode de protection qui peut prendre soit la valeur C (confidentialité et intégrité), I (intégrité seule) ou U (pas de protection),

Z est la valeur acceptée par B,

$K'$  est une clef de session pour un algorithme de chiffrement symétrique type DES,

Num est la valeur initiale des numéros de séquences des messages échangés qui seront utilisés par la suite dans le transfert.

1) A tire deux nombres aléatoires  $K'$  et Num

2) Message n :  $A \rightarrow B: A, B, \{Y, K', Num, H(Y, K', Num)\}_{K_b}$

3) B calcule

$Y, K', Num, H(Y, K', Num) = \{\{Y, K', Num, H(Y, K', Num)\}_{K_b}\}_{K_{b-1}}$  et vérifie le H.

4) Message n+1:  $B \rightarrow A: B, A, \{Z\}_{K_a}$

5) A calcule  $Z = \{\{Z\}_{K_a}\}_{K_{a-1}}$

I.4.1 A et B peuvent souhaiter des modes de protection différents. En fonction du mode de protection désiré par A et B, discutez la valeur finale de Z acceptée par B.

I.4.2 A quoi sert l'usage d'une clef de session? Pourquoi utiliser un protocole de type DES (et pas le RSA) pendant l'échange?

I.4.3 Pourquoi faut-il renuméroter les messages pendant l'échange alors que le protocole sous-jacent, qui est fiable, doit déjà le faire ?

1.4.4 Commentez les 5 étapes de l'échange précédent en expliquant la raison des différentes opérations

### 1.5 Phase de transfert

Proposez une structure des trames sécurisées échangées (les champs de contrôle qui doivent être ajoutés et le chiffrement appliqué), en fonction du mode de protection choisi.

### 1.6 Programmation en RPC

L'ouverture de connexion précédente est programmée dans un environnement d'appel de procédure à distance.

On structure l'application de la façon suivante. A et B échangent leur message en mode asynchrone sauf pour l'appel à S où A et B jouent le rôle de client et de serveur et S le rôle de serveur. On suppose que l'on dispose des primitives du niveau inférieur: ouverture\_connexion (émetteur, destinataire), envoyer(émetteur, destinataire, contenu), recevoir(émetteur, destinataire, contenu).

Décrire schématiquement les opérations réalisées sur le client et le serveur pour obtenir les clefs publiques.

## **Exercice 17 : Changement périodique de clés en cryptographie à clés publiques.**

On étudie un protocole de transmission d'informations développé pour sécuriser une voie point à point (logiciel de communication Nicecom). Dans l'une des versions on utilise un algorithme de cryptographie à clés publiques comme le RSA.

Chaque utilisateur dispose donc au départ d'un couple clé publique, clé privée pour un algorithme à clés publiques RSA:

Pour un utilisateur A: clé secrète DA et clé publique EA

Pour un utilisateur B: clé secrète DB et clé publique EB

Pour rendre excessivement difficiles les violations de sécurité on décide dans ce produit de pratiquer un changement périodique des clés de façon automatisée (par échange de messages en cours de dialogue).

En fait les clés vont donc être modifiées périodiquement de sorte que l'on utilisera pour un site comme A une suite de couples de clés:

$(EA(0) = EA, DA(0) = DA)$  , puis  $(EA(1), DA(1))$  , ....., puis  $(EA(i), DA(i))$  ,  $(EA(i+1), DA(i+1))$  , .....

A l'instant initial A utilise  $(EA(0), DA(0))$  et B connaît  $EA(0)$  et symétriquement. B utilise  $(EB(0), DB(0))$  et A connaît  $EB(0)$ .

On veut atteindre les objectifs suivants:

O1 : Authentification des correspondants.

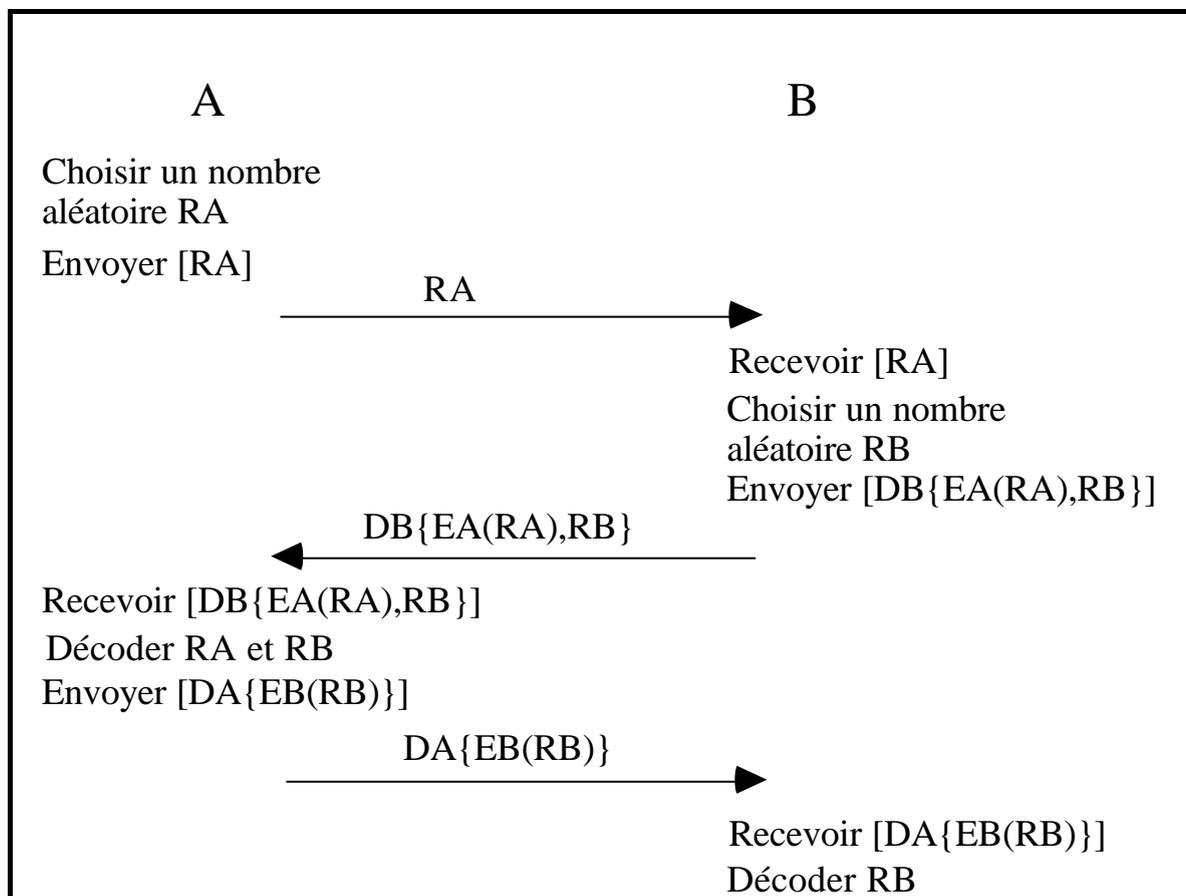
Dans les exemples traités A est l'appelant et B l'appelé.

O2 : Confidentialité des informations qui circulent sur le réseau.

O3 : Authentification de l'origine de tous les messages échangés.

O4 : Limitation de la durée d'usage des clés par changement périodique.

Le logiciel considéré utilise un protocole Px suivant entre les deux usagers A et B. Dans le schéma suivant on ne décrit pas complètement le rôle de tous les messages et l'utilisation des variables échangées (objet de questions du problème).



**Question 1 :** Quel type de problème permet de résoudre le protocole Px (justifiez votre réponse en discutant du fonctionnement du protocole Px en particulier du rôle des nombres RA et RB)? Que pensez vous de l'efficacité de ce protocole pour résoudre ce problème en termes de sécurité et de performances?

**Question 2 :** On prend le protocole Px comme base pour obtenir une version modifiée Py permettant le changement sécurisé des clés c'est à dire le passage du couple (EA(i), DA(i)) à (EA(i+1), DA(i+1)) et également du couple (EB(i), DB(i)) à (EB(i+1), DB(i+1)). On suppose que

les sites disposent d'algorithmes leur permettant de déterminer quand ils le souhaitent des couples clé publique, clé privée comme  $(EA(i+1), DA(i+1))$  pour l'algorithme RSA.

2.1 Proposez des modifications du protocole  $P_x$  pour définir le protocole  $P_y$  qui assurent le changement sécurisé des clés.

2.2 Quelles clés échange-t-on, à quel moment, selon quel cryptage? Justifiez votre solution. On rappelle que la sécurité de l'algorithme RSA repose sur le fait que les clés privées doivent rester confidentielles.

**Question 3 :** En quoi la technique de modification des clés appliquée rend-elle extrêmement difficile l'action des pirates.

3.1 On examinera d'une part le problème de décryptage par un pirate qui essaierait de casser le code par analyse des messages échangés.

3.2 On examinera ensuite le cas d'un pirate qui connaîtrait les clés initiales d'un utilisateur A ou B par des moyens autres que le décryptage (négligence de l'utilisateur). Que se passe-t-il?

3.3 L'accès au calculateur de l'émetteur ou du destinataire en session à distance constitue une possibilité d'attaque d'un tel protocole. Que doit faire un pirate qui a réussi à se connecter?

3.4 Quels sont les moyens matériels et logiciels qui protègent les systèmes de ce type d'attaque?

Questions 4:

4.1 Quand on est en phase  $i$  (avec les couples des clés  $(EA(i), DA(i))$   $(EB(i), DB(i))$ ) comment sont assurées la confidentialité et l'authentification des messages en mode RSA ?

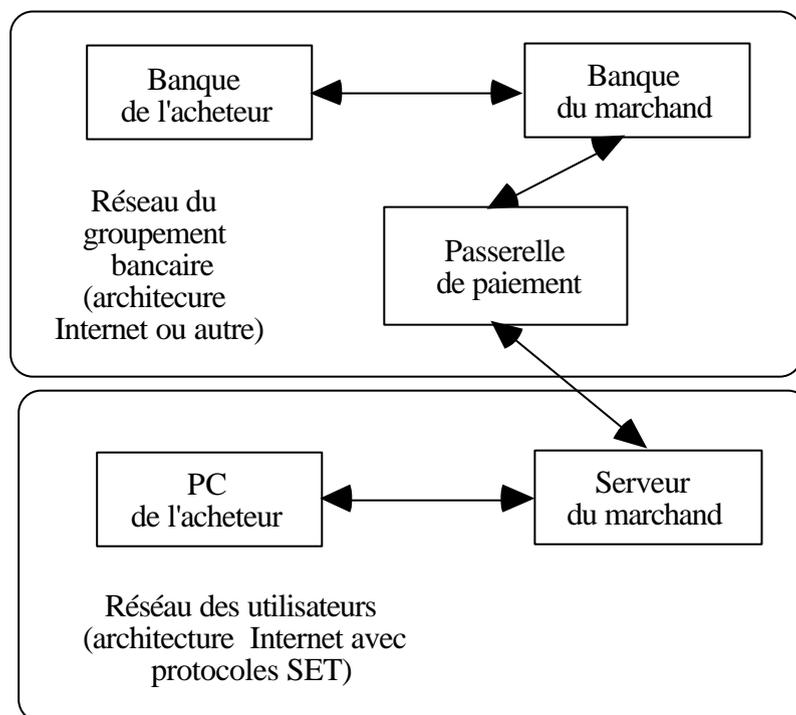
4.2 Quel est l'inconvénient majeur de cette approche ?

4.3 Comment pourrait-on résoudre ce dernier problème ?

## Exercice 18 : Commerce électronique sur Internet: SET ("Secure Electronic Transactions")

Pour sécuriser le commerce électronique sur Internet le standard SET a été proposé en 1996/1997. Il est soutenu par les principaux groupements de cartes bancaires Visa et Master Card ainsi que par de nombreux fournisseurs (IBM, Microsoft, ...). SET propose un ensemble de protocoles de sécurité pour le paiement par carte bancaire sur Internet. Il utilise pour cela des techniques de cryptographie (à clés publiques RSA, à clés secrètes DES, de fonction de hachage SHA-1). C'est une solution purement logicielle.

Une vision simplifiée de l'architecture de SET est donnée par la figure suivante où apparaissent les différents calculateurs de l'acheteur, du marchand, des banques ainsi qu'une passerelle faisant interface entre le monde Internet (utilisant le protocole SET) et le réseau bancaire.



Le fonctionnement de base est analogue à celui des cartes de crédits habituelles. L'acheteur connecte son poste de travail sur le serveur du marchand. Il consulte le catalogue des produits proposés, passe commande et autorise le paiement. Le marchand accepte la commande et la réalise. Le marchand pour se faire payer adresse à sa banque l'autorisation de paiement de l'acheteur via la passerelle de paiement. On trouve donc trois protocoles essentiels dans SET:

- Le protocole d'achat.
- Le protocole d'autorisation de paiement.
- Le protocole de paiement

Voici (parmi de nombreuses autres) quelques règles de sécurité de base que les protocoles SET doivent respecter :

a) L'acheteur et la passerelle de paiement doivent pouvoir vérifier que le marchand est bien celui qu'il prétend être. Le marchand doit pouvoir vérifier que l'acheteur et la passerelle de paiement sont bien ceux qu'ils prétendent être.

b) Une personne non autorisée ne doit pas pouvoir modifier les messages échangés entre le marchand et la passerelle de paiement.

c) Le marchand ne doit pas pouvoir accéder au numéro de la carte de l'acheteur.

d) Le banquier n'a pas à connaître la nature de la commande passée par l'acheteur au marchand.

1) Les problèmes de sécurité associés aux règles a) b) c) précédentes sont des problèmes généraux traités en cours. Donnez pour les règles a) puis b) puis c) les noms des problèmes associés. Rappelez de manière succincte la définition de ces problèmes (en une phrase).

SET est un protocole applicatif, qui définit une politique de sécurité. A partir des éléments précédents on cherche à spécifier cette politique

2) Quels sont les quatre rôles qui doivent être considérés ?

On définit les objets suivants

a) La carte bleue

b) Le PIN code de la carte bleue ("Personal Identification Number", le code secret)

c) Le numéro de la carte bleue

d) L'identifiant de la commande

e) Le contenu qualitatif de la commande (sa nature)

f) Le prix de la commande

Selon le cas on définit pour les objets précédents une ou plusieurs des méthodes suivantes:

A. Créer

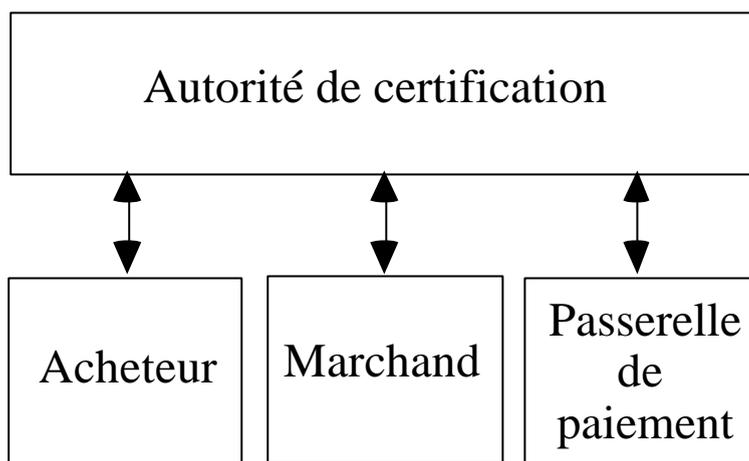
B. Lire (connaître la donnée)

C. Accepter (signer la donnée)

3) Pour chaque objet donnez les méthodes applicables (A, B ou C)

4) Etablir la matrice des droits: Il s'agit d'une matrice ayant en colonne (au nombre de 13) les méthodes et en ligne (au nombre de 4) les rôles. Si un rôle X a le droit d'utiliser la méthode y l'élément X,y est marqué à 1 et n'est pas marqué sinon.

Pour mettre en oeuvre les protocoles de sécurité utilisant la cryptographie SET définit une phase d'accréditation préalable des acteurs par une autorité de certification (en fait une hiérarchie d'autorités).



Vue globalement, l'autorité de certification délivre des certificats aux différents acteurs des protocoles SET.

5) Qu'est ce qu'un certificat? Quelles en sont les propriétés principales?

6) On étudie maintenant le processus d'achat. Il se déroule en deux échanges requêtes réponses successifs.

Premier échange (l'échange initial)

La requête initiale de l'acheteur vers le marchand indique simplement en clair l'intention par l'acheteur de passer commande.

La réponse initiale du marchand comporte trois éléments:

- un identifiant de commande plus sa signature numérique
- le certificat du marchand avec sa clé publique
- le certificat de la passerelle de paiement avec sa clé publique.

A la suite de ce premier échange, quelles vérifications peuvent être effectuées par l'acheteur ?

7) Le second échange du processus d'achat comporte l'envoi de la requête d'achat et une réponse d'accusé de réception de commande.

Envoi de la requête d'achat

L'acheteur construit la structure de donnée commande qui a vocation à être communiquée au marchand (produits, quantités, prix avec l'identification de la commande fournie par le marchand pendant l'échange initial...). Elle est baptisée par la suite OI ("Order Information").

L'acheteur construit la structure de données de paiement qui a vocation à être communiquée à la passerelle de paiement (informations concernant la carte bancaire de l'acheteur et identification de la commande à payer fournie par le marchand pendant l'échange initial). Elle est baptisée dans la suite PI ("Payment Information").

En fait les deux structures de données sont liées. Le paiement ne concerne que la commande identifiée. Il doit être effectué que si la commande est acceptée par le marchand. La commande n'est effective que si la banque approuve le paiement. De plus le contenu de la commande doit être caché à la banque et le contenu des instructions de paiement doit être caché au marchand.

Pour lier les deux structures de données, l'acheteur calcule par l'algorithme SHA-1 la fonction de hachage de chacune des structures de données SHA-1(OI) et SHA-1(PI). Il applique à nouveau la fonction de hachage SHA-1 à l'ensemble (SHA-1(OI), SHA-1(PI)) des fonctions de hachage concaténées. Il chiffre cette dernière empreinte en RSA avec sa clé privée. C'est en fait une signature numérique double qui est réalisée. Elle est baptisée dans la norme SET signature duale.

$$\text{Signature duale} = \left\{ \left\{ \{OI\}_{SHA1}, \{PI\}_{SHA1} \right\}_{SHA1} \right\}_{RSA}^{clef\_privée\_acheteur}$$

Le message suivant est préparé pour la passerelle de paiement:

PI, Signature duale

L'acheteur choisit une clé aléatoire clé\_aléa pour le DES. Le message a destination de la passerelle de paiement est chiffré en DES au moyen de cette clé.

$$\{PI, Signatureduale\}_{DES}^{clé\_aléa}$$

La clé DES est chiffrée au moyen de la clé publique de la passerelle arrivée avec le certificat de la passerelle.

$$\{clé\_aléa\}_{RSA}^{CLEF\_PUBLIQUE\_PASSERELLE}$$

Finalement le message de requête d'achat envoyé au marchand contient toutes les informations suivantes:

$\{PI, Signatureduale\}_{DES}^{clé\_aléa}$  ,  
 $\{PI\}_{SHA1}$  ,  
 $\{clé\_aléa\}_{RSA}^{CLEF\_PUBLIQUE\_PASSERELLE}$  ,  
 OI,  
 Signature duale,  
 Certificat de l'acheteur.

Envoi de la réponse du marchand à la requête d'achat

Le marchand construit un message de réponse qui a comme unique signification d'être un accusé de réception de la commande. Le marchand signe numériquement ce message (fonction SHA\_1 et chiffre RSA avec sa clé privée). Il ajoute à l'ensemble son propre certificat.

7.1) Comment le marchand vérifie-t-il l'intégrité de la commande OI?

7.2) Comment est réalisée la confidentialité des informations concernant la carte de crédit vis à vis du marchand?

7.3) Comment le marchand vérifie-t-il que l'acheteur est bien celui qu'il prétend être?

## Exercice 19 : Authentification des usagers et autorisation des requêtes dans le WEB

Le World Wide Web a tout d'abord été conçu comme un outil de diffusion de documents publics de sorte que peu d'efforts ont été effectués au départ pour contrôler l'accès aux informations offertes. Comme un domaine de plus en plus large d'informations et surtout de services ont été distribués au moyen du WEB, des outils de sécurité ont été proposés pour satisfaire les besoins qui sont apparus. Ce sujet examine des solutions successives qui ont été développées dans le cadre du protocole HTTP pour répondre aux besoins d'authentification des usagers et d'autorisation des requêtes qu'ils émettent.

La version 1.0 du protocole HTTP (HTTP/1.0 RFC 1945 mai 1996) propose un mécanisme de base de contrôle d'accès utilisant une authentification à mot de passe (mécanisme 'basic').

Si un utilisateur client requiert une page protégée d'un serveur WEB sans fournir de couple usager, mot de passe, il reçoit en réponse un code d'erreur 401 ('unauthenticated'). Sur réception de ce diagnostic le navigateur du client demande alors à l'utilisateur un nom d'utilisateur autorisé et son mot de passe au moyen d'un dialogue interactif dans une fenêtre. Lorsqu'une réponse est fournie le navigateur client réémet la requête vers le serveur avec les informations usager:mot\_de\_passe.

Lorsque l'on émet sur le réseau une requête avec le couple nom d'utilisateur et mot de passe, ces informations sont codées mais non cryptées (enregistrement de la requête baptisé 'Authorization'). La méthode de codage en format texte ascii employée est baptisée Base64. Elle consiste essentiellement à découper les informations par groupes de 6 bits et à représenter les groupes de 6 bits par un caractère ASCII. Par exemple si on souhaite transmettre le couple usager:mot\_de\_passe "Aladdin:open sesame", il apparaît dans la requête une ligne de la forme :

```
Authorization: Basic QWxhZGRpbjpvYVUHNlc2FtZQ==
```

De la sorte, si les mots de passe ne sont pas immédiatement lisibles dans les requêtes ('human readable'), il sont facilement décodables par un programme de décodage et peuvent donc être connus de tout le monde (les mots de passe ne sont pas cryptés).

1) Quels sont les avantages et les inconvénients en termes de sécurité et de coût de mise en œuvre d'une telle approche d'authentification ?

Le serveur WEB autorise une requête en consultant des fichiers créés par l'administrateur du serveur. Nous reprenons ici la protection de pages WEB par répertoires telle qu'elle est définie dans le cadre de l'outil NCSA Mosaic et reprise en version très voisine dans le serveur Apache. Le système d'exploitation est le système UNIX.

Supposons que l'administrateur d'un serveur WEB souhaite protéger un répertoire (baptisons le /mydir/turkey) contenant des pages WEB. Il doit créer dans ce répertoire un fichier dont le nom par défaut est .htaccess. Un exemple de fichier .htaccess est le suivant:

```
AuthUserFile /otherdir/.htpasswd
AuthGroupFile /dev/null
AuthName ByPassword
AuthType Basic
```

```
<Limit GET PUT>
require user daniel
</Limit>
```

Dans la première ligne de ce fichier on décrit où se trouve le fichier des mots de passe. Il est ici baptisé /otherdir/.htpasswd (c'est un fichier qui a pour nom .htpasswd et qui se trouve dans le répertoire otherdir). La seconde ligne indique qu'il n'existe pas de protection d'accès au niveau groupe d'utilisateurs, ce qui est indiqué par le fait que le fichier des protections de groupe est un flot vide (/dev/null). La chaîne associée à la troisième ligne (mot clé AuthName) peut être arbitrairement choisie par l'administrateur. Elle définit le nom du domaine auquel s'applique la politique de protection. Ce nom est transmis par le serveur au navigateur et il est affiché lors des demandes de mots de passe. Il permet à l'utilisateur de savoir quel nom d'utilisateur et quel mot de passe fournir (en fonction du domaine accédé). Ici le nom du domaine est un nom passe partout 'Bypassword'. La ligne 'AuthType' (type d'authentification) définit le protocole d'authentification comme étant 'Basic' c'est-à-dire celui que nous étudions ici. D'autres authentifications sont utilisables (PGP, KerberosV4, KerberosV5, ou Digest que nous examinons plus loin). On voit ensuite que dans ce fichier exemple seules les requêtes GET et PUT sont autorisées (enregistrement Limit GET PUT). On aurait pu définir ici une authentification pour d'autres opérations du protocole HTTP. Ici, l'autorisation est prévue uniquement pour l'utilisateur daniel.

Il faut bien sûr créer le fichier /otherdir/.htpasswd de mots de passe. Il contient des enregistrements de la forme usager:mot\_de\_passe. Les serveurs WEB disposent d'outils pour créer simplement par des dialogues interactifs ces fichiers de protection.

2) On suppose que l'administrateur du serveur WEB doit protéger d'autres pages WEB qui ont été créées dans un autre répertoire /mydir/goose. Ces pages appartiennent à un nouveau domaine baptisé realm. L'administrateur souhaite protéger l'accès à ces pages WEB en autorisant l'utilisateur daniel pour les opérations GET, PUT et POST, et l'utilisateur laurent uniquement pour les requêtes GET sur ces nouvelles pages. Quels sont les fichiers qui doivent être créés et à quel endroit doivent-ils se trouver?

Quand on réalise un service d'accès distant en réseau comme telnet ou de transfert de fichiers comme ftp, on commence par une ouverture de connexion ('login process') pendant laquelle on réalise une authentification de l'utilisateur. Cette authentification demeure effective pendant toute une période considérée comme formant un tout du point de vue de la protection. Pendant une telle session un utilisateur peut réaliser un ensemble d'opérations puis fermer la session quand bon lui semble.

3) Le protocole HTTP est-il conçu selon le principe précédent, c'est à dire que l'accès à un ensemble de pages, d'images ... d'un serveur forme un tout ou bien chaque accès est-il considéré comme indépendant des autres comme c'est le cas pour un serveur sans état (le serveur est-il avec ou sans état) ?

4) Quelles sont les conséquences du choix précédent relativement au problème d'authentification (quelle technique peut-on proposer côté navigateur client pour simplifier la vie de l'utilisateur à sa console) ?

A la version 1.1 du protocole HTTP (HTTP/1.1 RFC 2068 janvier 1997) est associée un autre protocole d'authentification baptisé « message digest authentication » défini par la RFC 2069 janvier 1997. Des améliorations ont encore été apportées au protocole de la version 'digest' par la RFC 2617 juin 1999.

On définit ici les principes généraux de la solution sans entrer dans les détails. Lorsqu'une réponse d'un serveur WEB à une requête d'un navigateur client est un rejet (code 401

'unauthenticated'), la réponse du serveur contient un champ particulier baptisé nonce qui contient une valeur aléatoire à la discrétion du serveur. Ce message de rejet devient alors ce que l'on appelle un challenge. Au lieu de répondre par le nom d'utilisateur et le mot de passe, le navigateur client doit alors répondre par un nom d'utilisateur et à la place du mot de passe la valeur :

MD5( concat ( nom d'utilisateur, mot de passe, nonce ) )

Dans l'expression précédente, concat désigne l'opérateur de concaténation. On voit donc que le navigateur ayant obtenu un nom d'utilisateur et un mot de passe doit fabriquer un texte qui concatène le nom d'utilisateur, le mot de passe et le nonce. Ensuite il lui applique la fonction MD5 qui définit une fonction de hachage à sens unique. Le résultat est la valeur transmise (à la place du mot de passe).

La RFC 2069 propose d'utiliser par défaut dans le protocole d'authentification de type 'digest' la fonction MD5 ('Message Digest version 5). D'autres fonctions de hachage à sens unique peuvent être négociées.

5) Rappelez la définition d'une fonction de hachage à sens unique.

6) Comment le serveur WEB réalise la vérification de l'autorisation d'accès à une page WEB protégée (le navigateur client ayant envoyé une requête avec nom d'utilisateur, MD5( concat ( nom d'utilisateur, mot de passe, nonce ) ) comme expliqué plus haut) ?

7) Pourquoi avoir appliqué la fonction MD5 au mot de passe et en même temps au nonce (qu'est ce que cela apporte en termes de sécurité) ?

8) La méthode de vérification permet-elle de stocker les mots de passe dans le fichier des mots de passe sous une forme cryptée (comme dans les fichiers de mot de passe d'accès à un système UNIX) ou bien doit-on avoir sur le serveur le fichier des mots de passe en clair ? Quelle est la conséquence relativement à la sécurité de la méthode?

La valeur du nonce dépend de l'implantation du serveur ('implementation dependent'). La norme suggère néanmoins pour assister les implanteurs de serveurs WEB une valeur de nonce qui peut-être recalculée, de manière à ce que la valeur obtenue soit identique (presque toujours) entre un challenge et sa réponse.

Nonce = MD5 ( concat (adresse\_IP, ":", estampille, ":", clé\_privée) )

Adresse\_ip est l'adresse IP de la machine du client.

Estampille est une valeur dépendante du temps qui ne change pas très souvent.

Clé\_privée est une valeur secrète du serveur.

9) Pourquoi choisir une estampille temporelle qui ne change pas très souvent ? Quel est le risque encouru par cette méthode?

10) Pourquoi avoir choisi d'introduire dans le nonce l'adresse IP du client, une estampille temporelle, un secret (en quoi le choix des valeurs utilisées dans le nonce limite t'il le risque encouru) ?

## Exercice 20 : S/MIME

S/MIME ('Secure/Multipurpose Internet Mail Extensions') définit un ensemble de standards pour transmettre de manière sécurisée des documents au format MIME.

1) Rappelez l'objectif poursuivi par la définition du format MIME. Citez deux grandes applications de l'Internet qui utilisent le format MIME.

S/MIME offre les grandes catégories de services de sécurité suivantes: authentification, intégrité, non-répudiation, confidentialité. Pour cela il permet de transférer de nouveaux types d'attachements qui sont des parties sécurisées. Une partie sécurisée est elle même une partie MIME (En tête + corps). On distingue trois types de parties dans le domaine de la sécurité pour lesquelles le standard définit un format précis (CMS 'Cryptographic Message Syntax' en S/Mime version 3):

Partie à signer ou à chiffrer ('data')

Partie chiffrée ('envelopped data')

Partie composant la signature d'un texte en clair ('siagned data')

2) A quoi sert chaque nouveau type ?

Un document signé est décrit en ASN1 par la définition suivante:

```
SignedData ::= SEQUENCE {  
  version Version,  
  digestAlgorithms DigestAlgorithmIdentifiers,  
  contentInfo ContentInfo,  
  certificates [0] IMPLICIT ExtendedCertificatesAndCertificates OPTIONAL,  
  crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,  
  signerInfos SignerInfos }
```

3) Que peuvent contenir et quels sont les usages des variables dont les types sont:

**DigestAlgorithmIdentifiers**

**ExtendedCertificatesAndCertificate**

**CertificateRevocationLists**

4) Lorsque l'on transmet un document chiffré il est possible de le compresser également. Dans quel ordre effectue-t-on les opérations: chiffrer, compresser, encoder en ASCII? (Justifiez votre réponse).

## Exercice 21 : Sécurisation des communications en Internet avec SSL-TLS

SSL ('Secure Sockets Layer'), est une norme de réseau qui permet de sécuriser les communications pour des applications Internet utilisant TCP/IP. SSL offre un service de communication analogue à celui des sockets mais SSL ajoute aux communications standards des fonctions de sécurité (authentification du client par le serveur, du serveur par le client, intégrité, confidentialité des données échangées) et éventuellement aussi des fonctions de compression.

Développé par Netscape jusqu'à la version 3.0 (novembre 1996), l'IETF a alors adopté SSL et a présenté sa version baptisée TLS ('Transport Layer Security' RFC 2246 en 1998) compatible avec SSL 3.0. Par rapport à SSL, TLS offre quelques extensions mineures comme une amélioration des signatures, différents traitements d'erreurs supplémentaires, ... En ce sens TLS 1.0 est parfois désigné comme SSL 3.1.

SSL/TLS est découpé en deux grandes parties. La partie 'Handshake' assure les fonctions initiales d'un échange sécurisé. La partie 'Record' assure les fonctions de sécurité sur les données utilisateur en appliquant des approches de cryptographie et de signatures définies pendant la phase de Handshake. Ce problème étudie plus particulièrement la partie Handshake.

La partie Handshake de SSL/TLS permet d'établir le contexte de sécurisation utilisé ensuite dans la phase d'échange de données. La partie Handshake permet l'authentification des entités communicantes. Elle permet également l'échange de clés de session.

Un contexte de sécurisation comporte :

- Un identifiant de session sécurisée choisi par le serveur.
- Un certificat d'entité distante (optionnel).
- Une méthode de compression (si la compression est appliquée).
- La suite cryptographique utilisée (voir plus loin).
- Une clé secrète ('master secret' 48 octets partagés entre client et serveur).
- Une variable indiquant si la session peut couvrir plusieurs connexions TCP.

Les opérations principales du protocole Handshake sont :

1. Négocier la suite cryptographique utilisée pendant le transfert des données.
2. Etablir une clé de session partagée entre le client et le serveur
3. Authentifier le serveur par le client (optionnel).
4. Authentifier le client par le serveur (optionnel).

En SSL/TLS une suite cryptographique est un choix relatif aux éléments suivants :

- La méthode d'échange de clés.
- La méthode de chiffrement utilisée pendant le transfert des données.
- La méthode de hachage utilisée pour la création d'une signature.

La méthode d'échange de clés peut se faire de deux façons. L'une utilise les algorithmes à clé publique et la notion de certificat. Une autre méthode est prévue en l'absence de certificats : la méthode de Diffie-Hellman.

Le chiffrement est réalisé au moyen d'un algorithme à clé secrète. Neuf algorithmes à clé secrètes avec des variantes sur les longueurs de clés sont possibles (DES, Triple-DES, IDEA, etc...)

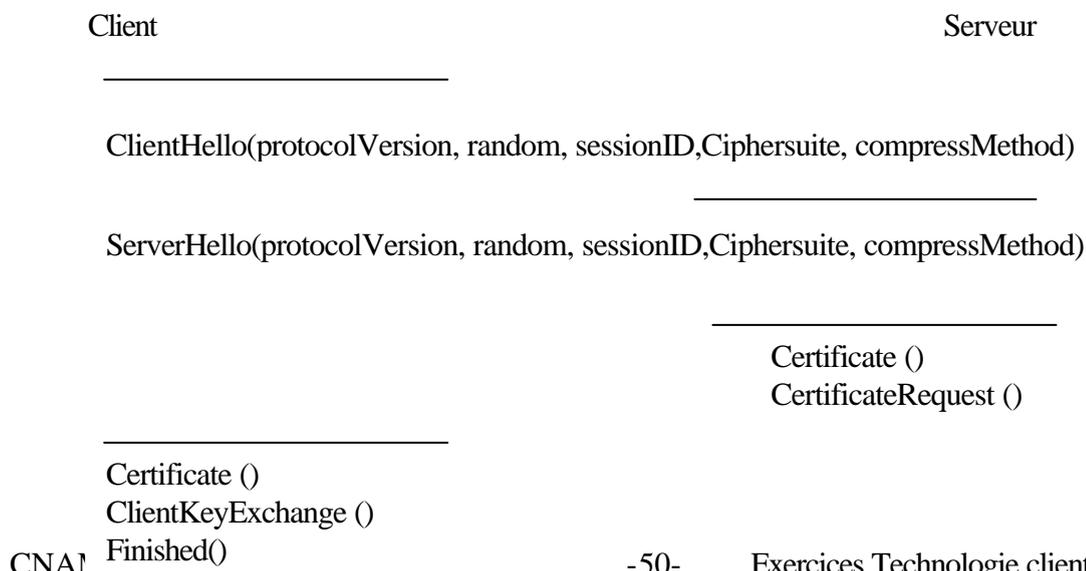
La fonction de hachage à sens unique (Digest Function) peut également être sélectionnée (MD5, SHA-1)

En combinant les différentes choix possibles dans les trois domaines précédents, la norme définit 31 suites de chiffrement cohérentes qui peuvent être adoptées après négociation.

- 1) Du point de vue du modèle OSI on considère que SSL-TLS est de niveau session alors que l'interface socket standard n'est pas de niveau session. Pourquoi ?
- 2) La plupart des utilisations de SSL/TLS comporte l'authentification du serveur par le client (bien que les fonctions d'authentification soient optionnelles). Citez une application de cette authentification. De manière générale pourquoi est il important d'authentifier un serveur ?
- 3) Pour un serveur, il est également possible avec SSL-TLS d'authentifier le client. Citez une application de cette authentification. De manière générale quelle est l'utilisation de cette authentification.
- 4) Le protocole SSL/TLS propose, dans l'une de ses modalités, d'utiliser la notion de certificat. Cette approche est d'ailleurs de loin la plus souvent retenue. A quoi sert un certificat. Rappelez les principaux champs d'un certificat ?
- 5) La vérification d'un certificat comporte différentes étapes. Quels sont les traitements à réaliser pour vérifier un certificat ?
- 6) Rappelez les principes d'une authentification en utilisant un algorithme à clés publiques ?

Les échanges du protocole Handshake sont assez complexes. En particulier ce protocole dépend des techniques de sécurisation utilisées (définies par le contexte de sécurisation négocié). On présente les principaux éléments du fonctionnement du protocole Handshake en omettant beaucoup de détails pour simplifier.

L'échange suivant est utilisé en cas d'authentification dans les deux sens au moyen de certificats. Ce protocole négocie le contexte de sécurisation, échange les certificats et les valide, construit un secret partagé sur 48 octets ('master secret') qui permet de fabriquer des clés de session et il échange des messages de terminaison.



---

Finished()

### Explications des échanges

1) La première phase (messages ClientHello, ServerHello) correspond à la négociation du contexte de sécurisation. Le client envoie différentes informations proposant un contexte de sécurisation (premier message avec version du protocole SSL, un nombre aléatoire, un nonce, une suite cryptographique, une méthode de compression). Le serveur choisit les valeurs définitives du contexte de sécurisation acceptable en fonction de la proposition client (second message). Il fournit également un nombre aléatoire.

2) Dans le cas d'une authentification du serveur, le serveur fournit son certificat (message Certificate). Il demande le certificat du client s'il y a aussi authentification du client (message CertificateRequest).

3) Le client valide le certificat du serveur.

4) Le client envoie son certificat au serveur. Il crée un secret au moyen d'un générateur de nombres aléatoires ('pre master secret'). Le client l'envoie au serveur encrypté avec la clé publique du serveur (message ClientKeyExchange). Le client génère le secret partagé (master secret) à partir du 'pre master secret' et des deux nombres aléatoires échangés dans les deux premiers messages.

5) Le serveur valide le certificat client. Il décrypte le secret (pre master secret) envoyé par le client au moyen de sa clé privée. Il génère selon le même algorithme que le client le même secret partagé (master secret).

6) En utilisant le secret maintenant partagé (master secret), le client et le serveur génèrent chacun de leur côté la même clé secrète de session utilisable pour des algorithmes à clés privées (comme le DES).

7) Le client et le serveur échangent des messages de terminaison (Finished) chiffrés au moyen de la clé secrète de session. Ces messages indiquent que le protocole de Handshake est terminé et que les échanges auront lieu à partir de maintenant en utilisant le contexte de sécurité négocié. Ces messages doivent être déchiffrés et vérifiés.

Question 7) Quels sont les mécanismes du protocole Handshake qui assurent l'authentification du serveur vis à vis du client?

8) Le protocole SSL utilise une combinaison de méthodes de cryptographie qui comprend des algorithmes à clés publiques et des algorithmes à clés privées. Pourquoi utiliser concurremment les deux techniques?

## Exercice 22 : Protocole de micro-paiement sur Internet

Dans ce problème Bob souhaite faire réaliser  $n$  opérations successives par Alice. Ces opérations ont la particularité d'être toutes les mêmes et d'avoir les mêmes paramètres. En fait nous allons utiliser dans la suite du problème ces opérations pour des paiements de sommes très faibles sur Internet (par exemple une somme fixe d'un centime d'euro à chaque opération).

Alice souhaite authentifier l'émetteur d'une opération (ici c'est Bob) à chaque opération.

1) Quels sont les avantages et les inconvénients d'une authentification unique en début d'une session (comme dans le protocole telnet) et d'une authentification pour chaque opération (authentification en continu) ?

Dans ce problème nous supposons que pour l'authentification en continu, Bob et Alice ont préalablement échangé une clef secrète  $C$  comportant  $k$  bits.  $H$  est une fonction de hachage de sécurité qui a été négociée entre Alice et Bob,  $\oplus$  est le ou exclusif,  $s$  est un nombre aléatoire sur  $k$  bits (nombre aléatoire de sécurité qui est différent à chaque authentification). On note  $\{C \oplus s\}^H$  le résultat de l'application de la fonction de hachage  $H$  au ou exclusif de  $C$  et de  $s$ . On note un message : émetteur, destinataire, type du message, données.

De manière à décomposer la solution on l'étudie progressivement. Dans une première étape on considère le cas d'une seule opération (donc  $n=1$ ). Le protocole appliqué est le suivant:

	Bob	Réseau	Alice
1	Bob génère un nombre aléatoire $s$ sur $k$ bits; il calcule $h_1 = \{C \oplus s\}^H$		
2		Bob, Alice, 'Transmission du nombre $s$ sur $k$ bits', $s$	
3	Bob demande à Alice la réalisation de l'opération	Bob, Alice, 'Demande opération', $h_1$	
4			Alice vérifie que $h_1 = \{C \oplus s\}^H$ Si oui Alice réalise l'opération

2) Quelles sont les propriétés des fonctions de hachage de sécurité ?

3) Pourquoi Estelle ne peut, en écoutant les messages qui circulent entre Alice et Bob, accéder à la clé secrète  $C$  partagée par Alice et Bob ?

4) Comment Alice peut-elle authentifier Bob ?

5) Comment s'appelle la technique de chiffrement utilisée dans ce protocole ?

On considère maintenant le protocole suivant défini pour n opérations successives :

	Bob	Réseau	Alice
1	Bob génère s aléatoire sur k bits et calcule : $h_0 = C \oplus s$ $h_1 = \{h_0\}^H$ ..... $h_i = \{h_{i-1}\}^H$ ..... $h_n = \{h_{n-1}\}^H$ Bob conserve la suite $h_1, \dots, h_i, \dots, h_n$		
2		Bob, Alice, 'demande préliminaire d'ouverture de n opérations', s	
3			Alice calcule la suite $h_0 = C \oplus s$ $h_1 = \{h_0\}^H$ ..... $h_i = \{h_{i-1}\}^H$ ..... $h_{n+1} = \{h_n\}^H$  Alice détruit les $h_i$ et ne conserve que $h_{n+1}$
4	Demande de la première opération	Bob, Alice, 'demande d'opération', $h_n$	
5			Authentification de Bob ? Réalisation de l'opération 1 si OK
...	.....	.....	.....
2+ 2i	Demande de la ième opération ( $i \leq n$ )	Bob, Alice, 'demande d'opération', $h_{n-i+1}$	
2+ 2i +1			Authentification de Bob ? Réalisation de l'opération i si OK

6) Bob mémorise toute la suite des  $h_i$  qu'il présente un par un à chaque opération en commençant par  $h_n$  puis  $h_{n-1}$ .... puis  $h_{n-i+1}$ .... Alice mémorise une seule valeur en commençant par  $h_{n+1}$  qu'elle remplace ensuite par  $h_n$  puis  $h_{n-1}$  et ainsi de suite à chaque nouvelle opération. Quelle est la vérification réalisée par Alice pour authentifier Bob aux étapes 5 et 2+2i+1 ?

7) Expliquez pourquoi Estelle, qui observe ce protocole ne peut à aucune étape en tirer des informations suffisantes pour pouvoir usurper l'identité de Bob.

Supposons que l'on se place dans une approche de cryptographie à clé publique. Alice et Bob disposent donc de couples clé publique, clé privée et de certificats accessibles dans un annuaire de certificats. Pour réaliser le protocole précédent d'authentification, dans une étape préalable, Alice et Bob doivent s'authentifier en utilisant un protocole à clé publique. Alice doit générer la clé secrète C et doit l'envoyer à Bob en utilisant également la méthode à clés publiques.

8) Donnez une solution pour cette authentification mutuelle et pour l'échange de clé C. Quel(s) protocole(s) de l'Internet pourrait servir à faire ces authentifications ?

Supposons que Alice soit une banque et Bob un de ses clients. On utilise maintenant le protocole précédent pour faire des micro paiements. Un protocole de micro paiement est un protocole permettant de débiter des sommes très faibles pour des applications de consultation payantes sur Internet (lire une page d'un document ou d'un journal par exemple). On suppose que n représente une somme en centimes d'euro ( $n = 1000$  centimes par exemple). Cette somme est débitée du compte courant de Bob au début et créditée un compte "de micro paiement" utilisable par Bob.

On suppose que chaque débit de micro paiement vaut un centime, ce centime est débité du compte de micro paiement de Bob lors de la réalisation par Alice d'une opération. Chaque opération du protocole précédent est donc un paiement d'un centime.

Quand le compte de micro paiement est vide, Bob doit alimenter à nouveau son compte de micro paiement et Alice et Bob recommencent toute une autre série d'opérations de micro paiement. Si Bob décide de ne plus payer de cette façon, alors que son compte de micro paiement n'est pas vide, son compte courant est crédité du solde du compte de micro paiement.

9) Expliquez pourquoi Bob ne peut pas faire de la fausse monnaie (chaque centime est identifié, ne peut être rejoué).

10) Expliquez pourquoi Estelle ne peut se substituer à Bob pour utiliser le compte de micro paiement de Bob.

11) Quel est l'avantage de ce protocole par rapport à un paiement qui utiliserait à chaque étape une authentification selon les méthodes habituelles (par exemple utilisation de SSL pour les paiements par carte bancaire sur Internet).

## Exercice 23 : Sécurisation du DNS : les normes DNSSEC-1

De nombreuses attaques dans le réseau Internet sont conduites en utilisant les faiblesses en matière de sécurité du DNS dans sa version de base (non sécurisée). En effet, cette version qui est encore pratiquement la seule utilisée ne possède aucun mécanisme de contrôle d'intégrité ou d'authentification. Par exemple, si un attaquant arrive à modifier les informations fournies par un serveur DNS ou par un cache DNS, il peut remplacer l'adresse d'une machine rendant un service normal par l'adresse d'une autre machine contrôlée par lui. Avec cette nouvelle machine qui va recevoir toutes les requêtes, un pirate peut réaliser différents types d'attaques (espionnage, déguisement, attaque par le milieu).

L'un des problèmes essentiels à tous les niveaux dans le DNS est donc le maintien de l'intégrité des informations gérées et transmises par les différentes entités DNS. C'est l'objet de propositions réunies sous le nom d'ensemble DNSSEC (pour DNS SECURITY extensions). Les normes DNSSEC sont encore en évolution. Nous étudions dans ce problème leur première version.

- 1) Dans le DNS rappelez le rôle des différentes entités communicantes suivantes: serveur primaire, serveur secondaire, cache et résolveur ?
- 2) Rappelez les grandes lignes des différents protocoles de communications que le DNS définit entre les différentes entités précédentes : serveur primaire, serveur secondaire, cache et résolveur.

### TSIG ('Transaction Signature' RFC 2845)

L'une des solutions, proposée pour la sécurisation du DNS dans le cadre DNSSEC, est baptisée TSIG. TSIG s'applique aux messages échangés par les protocoles DNS en définissant une signature ajoutée à ces messages. TSIG permet d'authentifier l'entité DNS émettrice et permet de signer en intégrité des données fournies par cette entité DNS en utilisant une fonction de hachage et une méthode de cryptographie symétrique (à clé secrète).

Pour les besoins du protocole TSIG, on rajoute deux informations dans chaque message émis. Tout d'abord on rajoute la date et l'heure de l'émission du message (chaque message est horodaté au moyen de l'horloge temps réel de la machine émettrice). On rajoute surtout dans chaque message protégé, une signature calculée au moyen de la clé secrète et d'une fonction de hachage sécuritaire selon la méthode HMAC sur laquelle nous revenons plus loin.

La RFC recommande d'utiliser une clé secrète par paire de communicants. Pour distinguer les différentes clés utilisées, chaque clé possède un nom logique qui doit être connu de l'émetteur et du destinataire. Ce nom est transmis dans chaque message sécurisé afin de déterminer la clé utilisée à l'émission et à utiliser par le récepteur.

- 3) Compte tenu des principes généraux de fonctionnement de TSIG qui viennent d'être décrits, entre quelles entités communicantes dans le DNS le protocole TSG vous semble t'il le mieux adapté (justifiez votre réponse) ?
- 4) Pourquoi le protocole TSIG a-t-il prévu un horodatage des messages au moyen de la date et de l'heure d'émission de chaque message? Quel contrôle est effectué par un récepteur de message? Expliquez pourquoi le champ d'horodatage doit être signé ?

5) Quelles propriétés doivent vérifier les horloges de chaque entité DNS pour que les contrôles d'horodatage soient efficaces? Quel protocole complémentaire à TSIG doit on absolument mettre en œuvre pour que l'horodatage fonctionne correctement ?

De manière générale on appelle en sécurité MAC (Message Authentication Code), une information rajoutée à un message qui sert de signature en intégrité en utilisant une fonction de hachage et une clé secrète. HMAC signifie **Keyed-Hashing Message Authentication Code**. HMAC est une méthode particulière de construction d'un MAC. HMAC est défini par la RFC 2104. C'est une méthode de signature assez largement utilisée (IPSEC, SSL, DNSSEC).

Pour construire une signature HMAC on utilise une clé secrète notée K et une fonction de hachage de sécurité notée H. Il peut s'agir par exemple de MD5 ou de SHA-1. De la sorte on a des HMAC-MD5 ou des HMAC-SHA-1.

Comme H n'est pas unique on note B le nombre d'octets généré pour chaque hachage (par exemple avec des condensés de 128 bits on a donc B=16 octets).

Si la clé K est trop courte (elle fait moins de B octets) on la complète avec des 0. Si la clé K est trop longue elle est tronquée à B octets par application de la fonction H.

On définit deux chaînes de B octets utilisées pour faire du bourrage ('padding'):

Ipad = 0x36 répété B fois (l'octet 36 en hexadécimal répété B fois).

Opad = 0x5C répété B fois (l'octet 5C en hexadécimal répété B fois).

La signature HMAC pour un message M est donnée par:

$HMAC(M) = H ( K \text{ XOR } Opad \parallel H ( K \text{ XOR } Ipad \parallel M ) )$

Dans l'expression précédente, XOR est l'opérateur de ou exclusif. La double barre  $\parallel$  indique la concaténation.

De manière à expliciter autrement la signature HMAC on peut encore dire qu'elle est obtenue au moyen des étapes suivantes de calcul:

- A. Si K fait plus de B octets calculer  $K = H(K)$ .
- B. Si K fait moins de B octets ajouter des 0 à K de façon à ce que K soit de longueur B.
- C. Calculer  $K \text{ XOR } Ipad$ .
- D. Concaténer le message M au résultat de 3.
- E. Appliquer H au résultat de 4.
- F. Calculer  $K \text{ XOR } Opad$ .
- G. Concaténer le résultat de 5 au résultat de 6.
- H. Appliquer H au résultat de 7.

6) La clé K est une clé secrète. Quelles précautions doit-on prendre concernant sa longueur, sa génération et son stockage?

7) Sur quelles propriétés de H reposent la sécurité du mécanisme de signature en intégrité HMAC?

8) Rappelez la méthode classique de signature vue en cours? Quel est l'avantage de HMAC en terme de performance par rapport à la méthode classique de signature?

### **Nouveaux enregistrements ressources KEY, SIG (RFC 2535 à 2539)**

Une autre solution proposée dans le cadre DNSSEC pour l'authentification et l'intégrité repose sur la création de deux nouveaux types d'enregistrements ressources (KEY et SIG) qui peuvent être stockés dans une base de données DNS.

L'enregistrement KEY a pour objectif de stocker des clés selon un format propre au DNS. Comme l'utilisation de clés ne concerne pas uniquement le DNS, l'enregistrement KEY peut aussi servir pour d'autres protocoles de sécurité (au niveau réseau avec IPSEC, au niveau transport avec TLS ou au niveau application avec SMIME).

La structure de l'enregistrement KEY est la suivante :

{Nom , TTL, Classe=IN, Type=KEY, (Indicateurs, Protocole, Algorithme, Valeur de clé)}

- Un nom d'enregistrement KEY peut appartenir à différents types permettant différentes utilisations de la clé. Dans le cas le plus usuel (sécurisation du DNS), il s'agit d'un nom de zone DNS pour lequel on définit une clé (par exemple cnam.fr). Un nom d'enregistrement KEY peut également être un nom de machine ou un nom d'utilisateur.
- Les indicateurs ('flags') définissent différents champs précisant l'utilisation de la clé (quel est le type du champ nom, ...)
- Le champ protocole définit le protocole de sécurité utilisant la clé (par exemple 3 : DNSSEC, ...)
- Le champ algorithme définit l'algorithme de chiffrement auquel est destinée la clé (par exemple 3 : DSA ...).
- La valeur de la clé (codée au format base 64).

L'enregistrement ressource SIG est placé après un enregistrement ressource donné. Il contient une signature pour cet enregistrement ressource. La technique de signature est la technique habituelle (ce n'est pas une signature HMAC). De la sorte, on peut interroger un serveur DNS sur un enregistrement ressource (mode de fonctionnement de base) mais on peut aussi demander la signature de cet enregistrement ressource. Pour améliorer les performances on peut aussi enregistrer une signature pour un ensemble d'enregistrements ressources.

La structure de l'enregistrement SIG est la suivante :

{Nom , TTL, Classe=IN, Type=SIG, (Type de l'enregistrement signé, Algorithme, TTL d'origine, Date d'échéance, Date de signature, Empreinte de la clé, Autorité signataire, Signature)}

- Type de l'enregistrement signé (par exemple SOA).
- Champ algorithme (définit les algorithmes utilisés pour générer une signature).
- TTL d'origine (définit le TTL de l'enregistrement signé).
- Date d'échéance (la date limite d'utilisation d'une signature).
- Date de signature (définit la date de création de la signature).
- Empreinte de la clé (définit un résumé, une empreinte de la clé utilisée).
- Autorité ayant signé (définit le nom de l'autorité qui a généré la signature).
- Valeur de la signature (codée au format base 64).

9) A quel type d'algorithme de chiffrement est selon vous destiné l'enregistrement KEY.

10) Avec les nouveaux enregistrements KEY et SIG comment se passe une requête d'accès sécurisée dans le DNS?

11) Les enregistrements KEY sont des enregistrements ressources qui peuvent être protégés aussi par un enregistrement SIG ? A quoi sert l'enregistrement SIG dans ce cas ?

12) Une idée qui a été proposée pour vérifier une clé est d'utiliser la hiérarchie de nommage du DNS comme hiérarchie des autorités de certification. Selon cette approche comment fonctionnerait alors la vérification d'une clé (par exemple si un enregistrement est sous l'autorité d'une zone iie.cnam.fr, quels sont les niveaux d'autorités impliqués dans le contrôle de cet enregistrement, quels sont les contrôles réalisés par le destinataire)?

13) Les champs date d'échéance de signature et date de signature de l'enregistrement montrent que la validité d'une signature dépend du temps. Pourquoi? Quel protocole est indispensable pour valider correctement les signatures ?



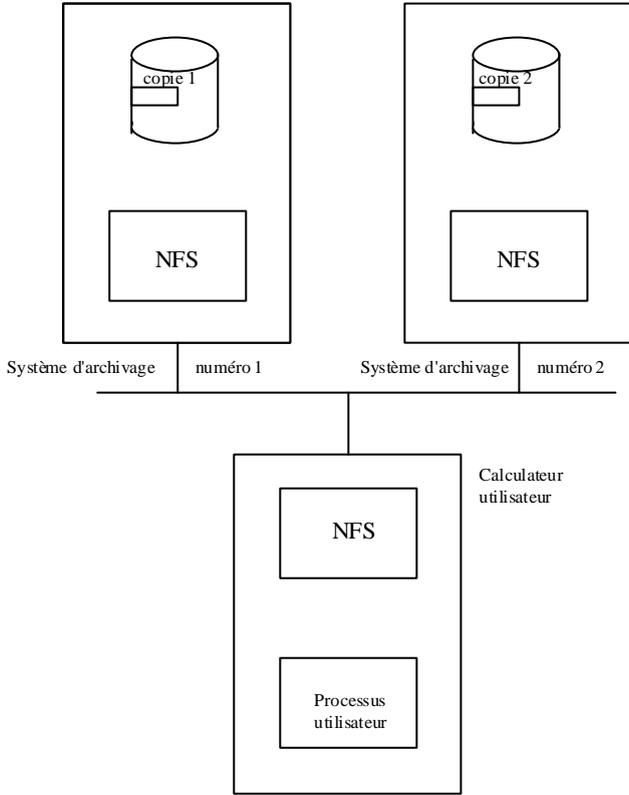
# TROISIEME CHAPITRE

## EXERCICES APPLICATIONS REPARTIES

### 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

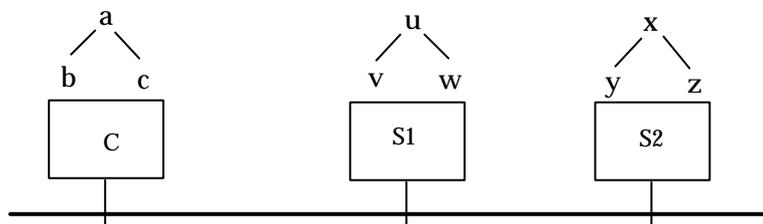
Proposez une méthode de réalisation de l'opération d'écriture qui utilise une diffusion de la requête à tous les systèmes d'archivages.

## 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 machines et accessible par l'utilisateur 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 :



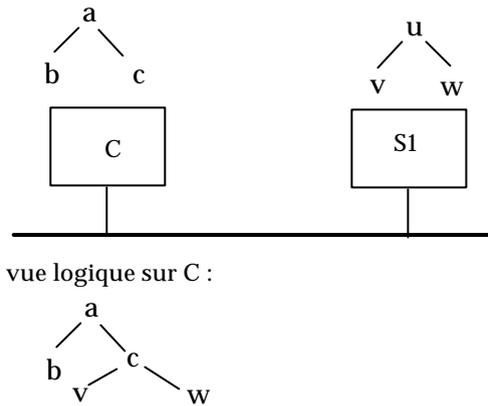
On dispose des opérations suivantes :

. 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.

### Exercice 3 : Messagerie X400

**Question 1** Rappelez l'organisation de la messagerie X400.

Quel est le rôle des différents modules :

- \* UA (User Agent) - Agent utilisateur
- \* MS (Message Store) - Mémoire de messages
- \* MTA (Message Transfert Agent) - Agent de transfert de messages.

**Question 2** Donnez les principaux champs ou attributs d'une adresse X400 que vous connaissez.

**Question 3** Dans la suite on trouve une trace de fonctionnement d'une messagerie X400. Donnez l'origine des messages ci-après et l'ensemble des pays traversés. La partie importante d'un message est **mise en gras**

a) message reçu sur une machine fonctionnant avec le système VMS

DUPONT (ATLAS@METIS) opened 20-AUG-1992 11:22:24.92  
Message 3 blocks

**From:**

**(C=fr,ADMIN\_DOM=atlas,PRIVATE\_DOM=cnam,ORGA=cnam400,SURNAME=dupont)**

**From Telephone:**

**From Location:**

Subject : TESTS X400

This Recipient Type : To Alternate Recipient : No

Return Contents : Yes Disclose Recipients : No

IP MessageID : A.098738

UA Content ID :

Message ID : 11711102802991/117002@EAGLE11=FR,2=ATLAS,3=CNAM

In Reply To :

Deliver Date : 20-AUG-1992 11:22:24.92 Expiry Date : 19-SEP-1992 11:17:10.00

Posted Date : 20-AUG-1992 11:20:54.00 Reply by :

Priority : First\_Class Receipt req'd : Yes

Sensitivity : Personal Reply request : Yes

Importance : High Delivery req'd : Full

Auto Forwarded: No Prohibit Conv : No

Obsoletes :

Cross Ref's :

To :

1 (DURAND@AM@VCNAM)

2 (NGUYEN@AM@PALLAS)

CC :

BCC :

Authorizing Users :

Reply To Users :

DEUXIEME ESSAI X400, LE PREMIER A ECHOUE

CAUSE MESSAGE NON REMIS.

b) Message reçu sur une machine fonctionnant avec le système Unix, ce message est passé par plusieurs pays :

From osimis-request@cs.ucl.ac.uk Sat Aug 29 00:44:28 1992  
**X400-Received: by /PRMD=cicb/ADMD=atlas/C=FR/;**  
**Relayed; 29 Aug 92 00:37:06+0200**  
**X400-Received: by /PRMD=uk.ac/ADMD=\_/C=gb/;**  
**Relayed; 28 Aug 92 22:42:39 GMT**  
**X400-Received: by /PRMD=uk.ac/ADMD=gold\_400/C=gb/;**  
**Relayed; 28 Aug 92 20:52:27 GMT**  
**X400-Received: by /PRMD=Hughes/ADMD=MCI/C=us/;**  
**Relayed; 28 Aug 92 13:51:46-0700**  
**X400-Received: by /PRMD=Hughes/ADMD=MCI/C=us/;**  
**Relayed; 28 Aug 92 13:41:03-0700**  
**X400-Received: by /PRMD=UK.AC/ADMD=GOLD\_400/C=GB/;**  
**Relayed; 28 Aug 92 13:40:56-0700**  
**Priority: Non-Urgent**  
**Date: 28 Aug 92 13:40:56-0700**  
**From: Danny J. Mitzel <dmitzel@whitney.hac.com>**  
Message-Id: <9208282041.AA11915@whitney.HAC.COM>  
To: osimis@cs.ucl.ac.uk  
Subject: MO Polling  
Importance: Normal  
Status: OR

Hi all,

Idanny (dmitzel@whitney.hac.com)

## Exercice 4 : Transactionnel réparti OSI-TP

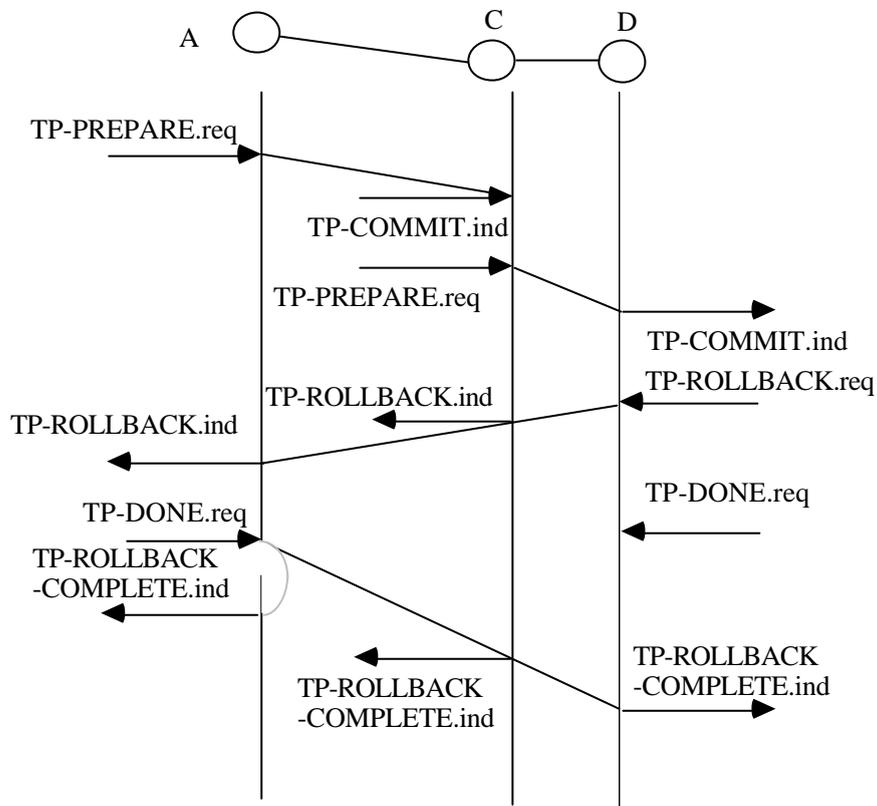
1 Rappelez les principaux problèmes à résoudre pour réaliser des transactions en univers réparti.

2 Le transactionnel réparti OSI-TP ('Transaction Processing') définit les notions de dialogues et de transactions. A quels besoins correspondent ces deux notions.

3 Les dialogues comme les transactions sont organisés en arbre. Pourquoi?

4 Quel est le principe général de la validation des transactions appliqué par OSI-TP ?

5 Commentez les échanges réalisés dans le diagramme suivant donné en cours.



## Exercice 5 : Transactionnel réparti

On suppose qu'une transaction de réservation de billets d'avion pour un parcours complexe en deux vols nécessite, à partir du calculateur émettant la transaction (calculateur d'une agence de voyage), l'accès à deux centres de réservation différents gérés par deux calculateurs S1 et S2. Par exemple la première partie du voyage concerne une compagnie intérieure puis la seconde partie est un voyage international sur une compagnie ayant un autre système de réservation. Le billet n'a de sens que si les deux vols souhaités sont obtenus simultanément.

La procédure appliquée est une approche transactionnelle répartie avec validation à deux phases. Dans la première phase sont réglés les problèmes de contrôle de concurrence et de préparation en mémoire stable de l'opération. Si celle-ci est validée (possible complètement) l'écriture définitive est réalisée sur les deux systèmes.

On suppose pour simplifier qu'une réservation commence par une lecture d'enregistrement  $tlire(x)$  permettant de savoir si des places d'un type  $x$  sont disponibles (par exemple classe affaire, fumeurs, coté fenêtre) suivie d'une opération  $préécrire(x)$  annonçant l'intention de réserver l'une des places disponible. Finalement une transaction peut se résumer:

Sur S1

$tlire(x)$ ,  
 $préécrire(x)$

Sur S2

$tlire(y)$   
 $préécrire(y)$

On suppose que par hasard deux agences émettent au même instant la même transactions de réservation (notées dans la suite  $Ta$  et  $Tb$ ) et que l'entrelacement des opérations amène à constater l'ordre suivant des opérations sur chacun des calculateurs (l'ordre est indiqué par des numéros locaux à chacun des calculateurs).

$Ta$	Sur S1	Sur S2
$tlire(x)$	1	
$técrire(x)$	3	
$tlire(y)$		1
$técrire(y)$		2

$Tb$	Sur S1	Sur S2
$tlire(x)$	2	
$técrire(x)$	4	
$tlire(y)$		Non défini
$técrire(y)$		Non défini

On utilise une technique de contrôle de concurrence en univers réparti basée sur le verrouillage à deux phases.

**Q1** Rappelez le principe du verrouillage à deux phases.

**Q2** Décrivez ce qui se passe dans l'exemple précédent en phase de pose des verrous. Peut-on valider les transactions? Pourquoi?

**Q3** Comment assurer la validation des transactions  $Ta$  et  $Tb$ ?

**Q4** On implante les transactions précédentes au moyen de la norme OSI-TP. Donnez pour ces transactions la structure de l'arbre des transactions.

**Q5** Quels sont les éléments de service TP que l'on peut constater pour chaque transaction impliquant un calculateur client de réservation  $C$  et les deux ordinateurs serveurs de réservation  $S1$  et  $S2$ . Dessinez un diagramme à flèches comportant la description des éléments de services générés par les trois sites, leur séquençement dans le temps (on suppose que tout se passe bien et qu'un seul message de données est utilisé dans chaque sens pour les besoins d'une transaction).

## Exercice 6 : Système transactionnel réparti

On souhaite mettre en oeuvre un système de réservation pour des agences de voyages. Des fournisseurs de moyens de transport, des fournisseurs de moyens d'hébergement ainsi que des prestataires de services (sports, séminaires) s'associent dans ce projet. L'objectif est de fournir un service personnalisé aux clients qui souhaitent réserver en une seule fois dans une agence leur moyen de transport, leur hébergement et leurs activités pour un séjour.

Jusqu'à maintenant, l'hôtesse d'accueil de l'agence effectuait les recherches successives et vérifiait la compatibilité de la demande du client. On souhaite donc lui faciliter la tâche en équipant les différents participants (agences de voyages et fournisseurs de service) d'un système transactionnel réparti. Ainsi un client entrant dans une agence de voyages pourra obtenir une réponse rapide à sa demande. Tous les participants à ce nouveau service sont déjà dotés d'un matériel et de logiciels informatiques leur permettant de réaliser leur réservation. On souhaite donc les interconnecter via un service de traitement de transactions réparties.

1- La fiabilité d'un système transactionnel vient, entre autres, du fait que l'on utilise des mémoires stables. Rappelez ce qu'est une mémoire stable et donnez un exemple.

2- Le système de transactions mis en place autorise l'exécution parallèle de transactions. Rappelez la définition du principe de la sérialisation. Quels sont les problèmes qui se posent ?

3- Rappelez les règles de dépendances entre les opérations de lecture et d'écriture dans un système transactionnel. rappelez ce qu'est une transaction vivante (notée T) et une transaction validée (notée T\*)

4- Pour détecter les conflits éventuels, on construit un graphe de dépendance. Soient les transactions suivantes avec leurs opérations et leur ordre d'exécution :

T1	T2	T3	T4
4:préécrire(y)	3:lire(y)	2:préécrire(x)	1:lire(x)
5:lire(z)	6:préécrire(x)	7:préécrire(z)	8:valider
	9:valider		

- Donner le graphe de dépendance
- La cohérence est-elle maintenue ? Justifiez votre réponse

5- Soit la transaction suivante :

Un client souhaite réserver un voyage qui comprend un aller-retour Paris-Singapour, un hôtel 3 étoiles (pour une semaine) ainsi qu'un circuit pour les 5 jours suivants. Le vol se réserve auprès de Air-Singapour (Serveur X), l'hôtel auprès du centre de réservation de Singapour (Serveur Y) et le circuit auprès d'un tour operator spécialisé dans l'Asie (Serveur Z). La date souhaitée de départ est le 19-9-96.

- Donner l'ensemble des échanges protocolaire nécessaires pour réaliser la transaction par un protocole de validation à deux phases dans le cas où tout se passe correctement. Vous utiliserez les primitives offertes par OSI-TP

6- Trois types de pannes sont possibles dans le cadre d'un protocole:

- panne du client ou coordinateur
- panne des serveurs ou exécutants
- panne du réseau

Pour détecter les pannes, on utilise des délais de garde.

Définir les endroits stratégiques côté client dans le protocole de la question précédente où vous placez les délais de garde et les conséquences en cas de retombée de ces délais de garde. Proposez pour chaque cas une stratégie à adopter en cas de retombée du délai de garde

7 - Même question que précédemment, mais côté serveur

8- Suite à ces différentes pannes possibles, donner les faiblesses de ce protocole de validation à deux phases

## Exercice 7 : CORBA et le transactionnel

1- Les services dans un système d'objets répartis comme CORBA sont des ensembles d'objets qui assurent des fonctionnalités nécessaires à un grand nombre d'applications distribuées telles que le nommage, la persistance, la sécurité ou les transactions. Expliquez les fonctions réalisées par le service de nommage ("naming service")? Donner l'exemple d'au moins un autre service de nommage différent de celui de CORBA.

2- Corba définit un service de sécurité. Rappelez les principaux problèmes de la sécurité? Quelles sont les fonctionnalités que l'on peut trouver dans un service de sécurité d'un système d'objets répartis?

3- Le service de transactions de CORBA assure l'exécution de traitements transactionnels mettant en œuvre plusieurs objets. Ce service fournit essentiellement un ensemble de primitives pour la validation à deux phases. Rappelez brièvement les objectifs du protocole de validation à deux phases et les principes généraux de son fonctionnement. Citez les principales primitives utilisées pour l'une des interfaces vue en cours (par exemple Xopen DTP XA ou OSI TP)

4- On s'intéresse à la transformation du protocole de validation à deux phases en mode message (question précédente) en un service transactionnel pour un système d'objets répartis utilisant des invocations synchrones de méthodes distantes sur des objets CORBA.

En fait on cherche à mettre en correspondance des couples de messages requête/réponse avec des méthodes d'un service transactionnel invoquées sur des objets CORBA de façon synchrone. Soit les trois méthodes suivantes:

**Prepare (out Boolean resultat)**

Cette méthode permet de d'exécuter la première phase. Elle a deux réponses possibles : vraie si OK et faux si non OK.

**Validation ()**

Cette méthode exécute la seconde phase dans le cas de succès.

**Abandon ()**

Cette méthode exécute la seconde phase dans le cas d'échec.

Dans le protocole de validation à deux phases, quels sont les couples de messages requête réponse qui correspondent aux appels de procédure précédents. Expliquez brièvement votre réponse en rappelant la fonction des couples de messages et la fonction de chaque méthode

5- On considère maintenant un client A (coordinateur) souhaitant réaliser une transaction avec deux sites serveurs B et C (exécutants de sous transactions). Sur le serveur B, A souhaite écrire des données à l'aide la méthode Ecrire-Données(). Sur le serveur C, A souhaite réaliser un calcul à l'aide de la méthode Calcul(). Le client A utilise pour cela un objet réalisant un service transactionnel pour CORBA utilisant les méthodes définies à la question précédente et différentes primitives comme:

Debut\_Transaction et Fin\_transaction qui délimitent une transaction.

Parbegin et Parend permettent d'activer en parallèle plusieurs procédures distantes afin de faire travailler en parallèle plusieurs serveurs afin d'améliorer les performances (ici pour les deux serveurs B et C).

La structure du pseudo-code du client A est fournie en annexe. Ajoutez sur la feuille les primitives de validation à deux phases permettant de mettre en œuvre cette transaction côté client.

6- Automate. Donnez une représentation sous forme d'un automate du pseudo-code du client faisant l'objet de la question 5.

7 Gestion des défaillances. Le code précédent ne fait pas apparaître de délais de gardes. On s'intéresse aux pannes franches des serveurs qui réalisent les sous transactions. Rappelez les principes de la solution pour détecter et gérer les pannes franches des serveurs dans le protocole de validation à deux phases?

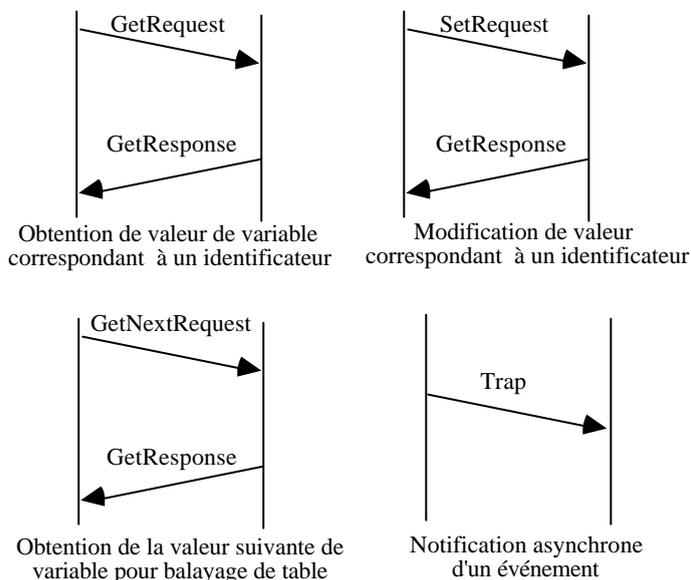
8 - On souhaite représenter sur un automate le mécanisme de traitement des pannes des serveurs. Ajoutez sur l'automate de la question 6 les éléments qui correspondent au traitement de la panne franche des serveurs: délais de garde, traitements à échéance ou l'on suppose que le serveur est en panne (représentez les états, les transitions avec événements déclencheurs (?) et actions (!))

## Exercice 8 : 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 de 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 baptisée 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 (IPaddress) 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 addresses
                                -- IP address 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
    }

```

```

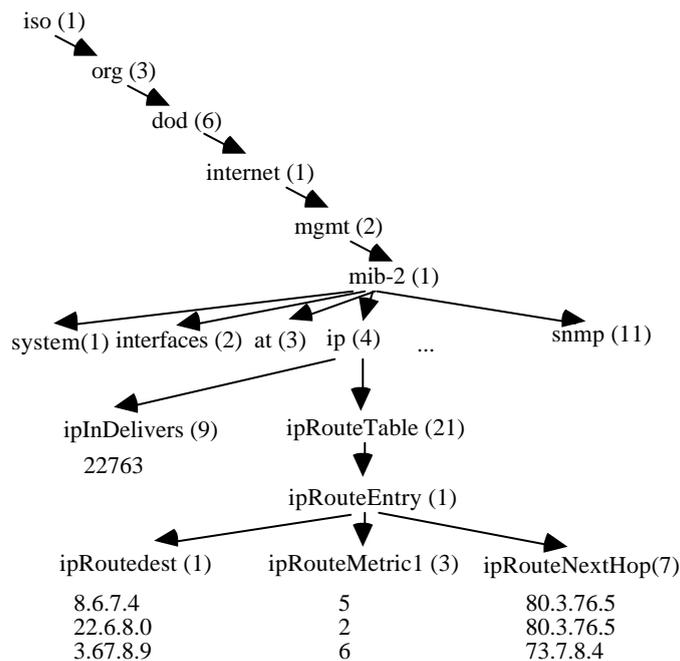
VarBind ::= SEQUENCE      - Variable binding
          {name ObjectName,
           value ObjectSyntax }
VarBindList ::= SEQUENCE OF VarBind
END

```

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?

## Exercice 9 : Échange de données informatisées EDI

### 1 Notions générales

Les questions suivantes n'appellent pas en réponse la recopie complète d'un chapitre de livre ou de polycopié. Il est demandé de faire une synthèse brève (d'une dizaine de lignes au maximum) qui rassemble les idées principales correspondant à la question.

**1.1 Quels sont les objectifs assignés par ses concepteurs à l'EDI et quels en sont les moyens principaux?**

**1.2 Quels sont les moyens de communication (physiques ou architectures de réseaux) sur lesquels il est prévu que ce protocole fonctionne?**

**1.3 Quels sont les principes généraux retenus pour le codage des informations échangées qui permettent l'interopérabilité des applications?**

**1.4 L'EDI définit une syntaxe de transfert différente de celle de ASN1. Quels sont les avantages et les inconvénients des deux approches?**

**1.5 Quelles sont selon vous les forces et les faiblesses de l'EDI?**

### 2 Étude d'un format d'échange EDI

En cours de mise au point d'un logiciel EDI on intercepte les informations suivantes codées en EDIFACT:

```
UNB+UNOA:1+56789:12+12345:91+950905:1445+REF34+4287+INVOIC'  
UNH+INV001+INVOIC:90::UN'BGM+380+67-066-Y-445324+950905'  
NAD+SU+5678912+ +SOCIÉTÉ DE MÉCANIQUE INDUSTRIELLE+15 AVENUE  
POINCARÉ+COLOMBES+ +92700+FR'
```

Dans cette technique de codage EDI les informations minimum qui doivent figurer comportent d'une part:

- un identifiant de l'autorité administrative qui normalise et un numéro de version de la syntaxe employée,
- un identifiant de l'émetteur du message qui peut être suivi d'un attribut qualifiant cet identifiant,
  - un identifiant du destinataire (avec son attribut).
  - la date (AAMMJJ) et l'heure (HHMM) de la fabrication du document,
  - une référence unique de l'échange attribuée par l'émetteur,
  - des éléments facultatifs comme un mot de passe, le type de document, ...

Elles comportent d'autre part:

- le numéro d'identification du message,
- une description de la nature du message avec
  - . le type du document
  - . le numéro de version du type
  - . d'autres informations

**2.1 Définissez précisément le rôle respectif des caractères + (plus), : (deux points), ' (apostrophe)?**

**2.2 Quel est le rôle du segment UNB?**

**2.3 Quel est le rôle du segment UNH?**

**2.4** Le symbole BGM est une abréviation pour "BeGin Message". Il comprend le type codé numériquement d'un document, la référence commerciale d'une facture et sa date. Le symbole NAD est une abréviation pour "Name And Address". Il comprend l'indication de type de l'organisme ou la personne concernée (ici SU pour vendeur ou BY pour acheteur) puis l'identifiant puis l'adresse à définir.

**A partir de l'exemple et de toutes les informations en votre possession déterminez:**

**a) - le codage numérique du type de document facture**

**b) - l'heure de fabrication de la facture**

**c) - la référence de cette facture dans la nomenclature de l'émetteur**

**d) - son identifiant unique EDI**

## Exercice 10 : XML

XML ('eXtended Markup Language') est un langage de balisage ('markup language'), 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](mailto:expediteur@domaine)

To: [destinataire@domaine](mailto: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](mailto:copie@domaine)

Message-ID: [numero@domaine](mailto: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).

## Exercice 11 : 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 OSI 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

[ règle ]            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 Resource 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*(suite non vide de caractères)
segment = *(suite de caractères)
DIGIT = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 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, le résultat de scripts CGI et les applets Java sont chargés automatiquement 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 script CGI dont le résultat est une image au format GIF, un fichier son, 1 applet Java.

**8.1) Quel est le nombre de requêtes GET émises par le client vers le serveur?**

**8.2) Pourquoi les séquences vidéo ne sont-elles pas chargées automatiquement?**

9) Les balises HTML suivantes délimitent:

- une image fixe : `<IMG SRC = "...">`,
- une applet Java : `<APPLET CODE = "...">`
- le résultat affichable d'un script `<IMG SRC = "...">`

La commande de chargement 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 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.**

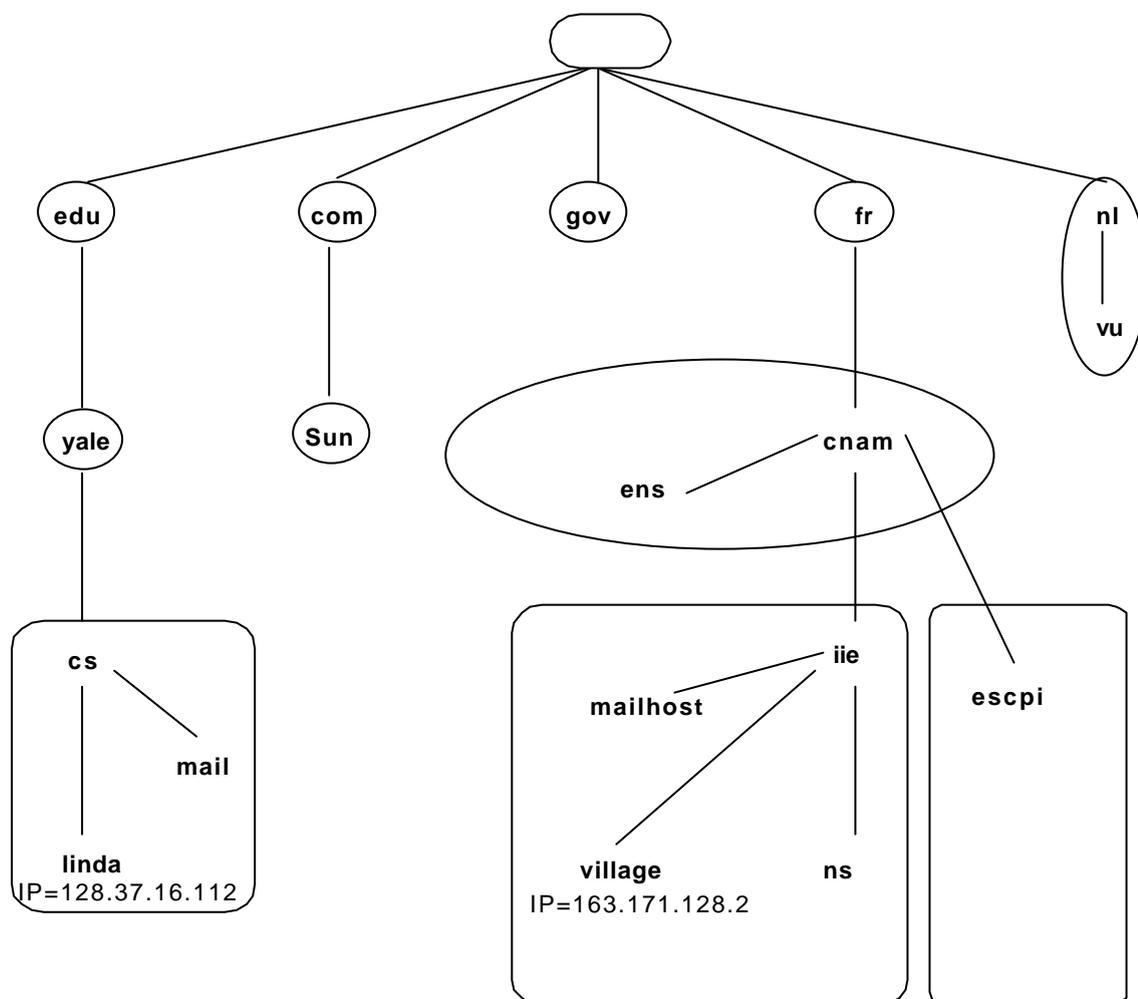
**12) L'utilisateur a modifié la configuration de son client navigateur de manière à ne plus utiliser de cache local client mais un cache serveur sur son site.**

**12.1) Comment se comporte maintenant un client?**

**12.2) Quel est l'intérêt de cette nouvelle organisation?**

## Exercice 12 : DNS "Domain Name System" sujet 1

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'utilisateur 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](mailto:superman@linda.cs.yale.edu) utilise la résolution itérative du DNS pour atteindre l'utilisateur [asterix@village.iie.cnam.fr](mailto: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 13 : DNS "Domain Name System" Sujet 2

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](http://www.discount.com)

6.17.173.163.in-addr.arpa

ulyse.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?

Le renforcement de la sécurité du DNS est un problème d'actualité qui a fait l'objet de plusieurs RFC. En effet un pirate peut par exemple remplacer l'adresse du serveur de nom d'un domaine par sa propre adresse (s'il a pris par exemple le contrôle d'un routeur). Il peut ensuite faire croire ce qu'il veut au résolveur en fabricant les réponses.

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 14 : Services et protocoles d'annuaires répartis

### Norme LDAP et sécurité des transactions Internet

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.

#### Question I.1

I.1.1 DNS "Domain Name System" est le service d'annuaire (ou de gestion de noms) de l'Internet. Rappelez quels sont les objectifs du DNS.

I.1.2 DAP 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.

#### Question I.2

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 1779 (un exemple simple est donné plus loin).

La PDU bindRequest 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 Kerberos ou de DSA ("Digital Signature Algorithm").

**I.2.1** Que signifient dans la description précédente les expressions APPLICATION 0, SEQUENCE, INTEGER, CHOICE?

Un client envoie une PDU bindRequest avec les valeurs suivantes:

```
version:: = 2
name:: = 'CN=James Hacker,L=Basingstoke,
         O=WidgetInc,C=GB'
authentification simple (sans mot de passe)
```

La zone name fournit ici un exemple de nom X500 ("distinguished name"). De nombreux attributs peuvent être précisés. Ici on a précisé CN pour CommonName, L pour Localityname, O pour Organisation, C pour Country. Les questions qui suivent concernent le codage des données échangées lors du transfert de la PDU bindRequest associée à l'exemple précédent.

**I.2.2** Rappeler les principes de la syntaxe de transfert associée à ASN1?

**I.2.3** Donnez octet par octet la valeur binaire ou hexadécimale du codage du champs "version" transmis du client vers le serveur. Vous expliquerez pour chaque octet la raison de ce codage et la valeur numérique obtenue.

**I.2.4** Donnez octet par octet la valeur binaire ou hexadécimale du codage du champs "name" transmis du client vers le serveur. Vous expliquerez pour chaque octet la raison de ce codage et la valeur numérique obtenue. Il n'est pas demandé de convertir la chaîne de caractère en binaire.

**I.2.5** Pour le codage de CHOICE on donne les indications suivantes. Le constructeur CHOICE n'est pas codé, mais on code directement la donnée choisie. On remarque qu'elle est étiquetée mais que cette étiquette n'est associée à aucun mot clé. Elle est donc spécifique au contexte. Le mot clé IMPLICIT n'est pas présent.

Donnez octet par octet la valeur binaire ou hexadécimale du codage du champs "authentication" transmis du client vers le serveur. Vous expliquerez pour chaque octet la raison de ce codage et la valeur numérique obtenue.

**I.2.6** Donnez la valeur hexadécimale (octet par octet) du codage du message complet transmis du client vers le serveur en expliquant pour chaque octet la raison de ce codage et la valeur numérique obtenue. Il n'est pas demandé de convertir la chaîne de caractère en binaire.

### Question I.3

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.

**I.3.1** 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?

**I.3.2** Rappelez la définition d'un automate de service.

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

**I.3.3** 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.

**I.3.4** 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.

## II - Sécurité Internet- Certificats X509

L'Internet utilise des certificats dans de nombreux protocoles de sécurité. Un certificat est une donnée liant un nom, une clef publique associée à ce nom et le tout est signé par une autorité de certification. Ces certificats ont un format défini dans la norme X509 et sont transmis via le protocole LDAP

La spécification suivante définit (partiellement) un certificat au format X509:

```
Certificate ::= SIGNED { SEQUENCE {
    version [0]                Version DEFAULT v1,
    serialNumber               CertificateSerialNumber,
    signature                  AlgorithmIdentifier,
    issuer                     Name,
    validity                   Validity,
    subject                    Name,
    subjectPublicKeyInfo       SubjectPublicKeyInfo,
    issuerUniqueIdentifier     [1] IMPLICIT UniqueIdentifier OPTIONAL,
    -- If present, version must be v2 or v3
    subjectUniqueIdentifier    [2] IMPLICIT UniqueIdentifier OPTIONAL
    -- If present, version must be v2 or v3
    extensions                 [3] Extensions OPTIONAL
    -- If present, version must be v3 -- }
}

Version ::= INTEGER { v1(0), v2(1), v3(2) }
CertificateSerialNumber ::= INTEGER
AlgorithmIdentifier ::= SEQUENCE {
    algorithm                  ALGORITHM.&id({SupportedAlgorithms}),
    parameters                 ALGORITHM.&Type ({SupportedAlgorithms}
    { @algorithm }) OPTIONAL }

SupportedAlgorithms ALGORITHM ::= { ... }
Validity ::= SEQUENCE { notBefore UTCTime, notAfter UTCTime }
SubjectPublicKeyInfo ::= SEQUENCE { algorithm AlgorithmIdentifier,
    subjectPublicKey BIT STRING }

Time ::= CHOICE { utcTime UTCTime,
    generalizedTime GeneralizedTime }

SIGNED { ToBeSigned } ::= SEQUENCE {
    toBeSigned ToBeSigned,
    encrypted ENCRYPTED { HASHED {ToBeSigned} }
}
```

## Question II.1

**II.1.1** Qu'est ce qu'un algorithme de signature numérique. Quelles en sont les propriétés?

**II.1.2** A quoi servent les champs suivants:

**serialNumber ,  
signature ,  
issuer,  
validity,  
subject,  
subjectPublicKeyInfo,  
issuerUniqueIdentifier**

**II.1.3** Expliquez la spécification ASN1: SIGNED { ToBeSigned } d'une donnée signé.

## Question II.2

Un serveur de certificats est accessible via LDAP pour obtenir des certificats X509. Supposons qu'une entité A veuille contrôler la validité de la signature d'un document signé par une entité B. Il dispose au départ de l'identifiant de B, de l'adresse du serveur de certificat et du certificat de l'autorité de certification qui a signé le certificat de B.

**II.2.1** Listez la suite des opérations que A doit réaliser pour contrôler la signature de B.

**II.2.2** Les certificats X509 sont maintenant fréquemment utilisés dans l'Internet. En considérant des applications extrêmement courantes de l'Internet, donnez deux domaines d'applications d'une telle procédure.

## Exercice 15 : 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 Middleware' 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.

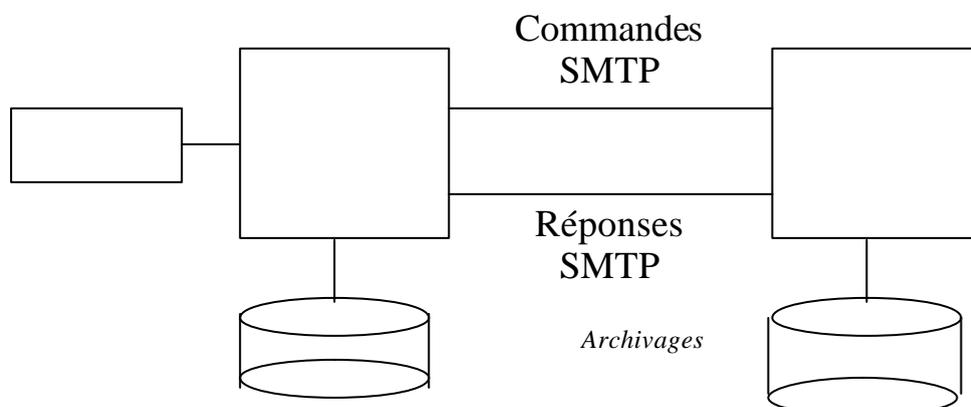
I.4) Il existe deux grandes catégories d'architectures pour les messageries. La première catégorie est basée sur un principe de délivrance de bout en bout ('end to end delivery'). La seconde catégorie est basée sur une approche de stockage et retransmission ('store and forward delivery'). Rappelez les 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?

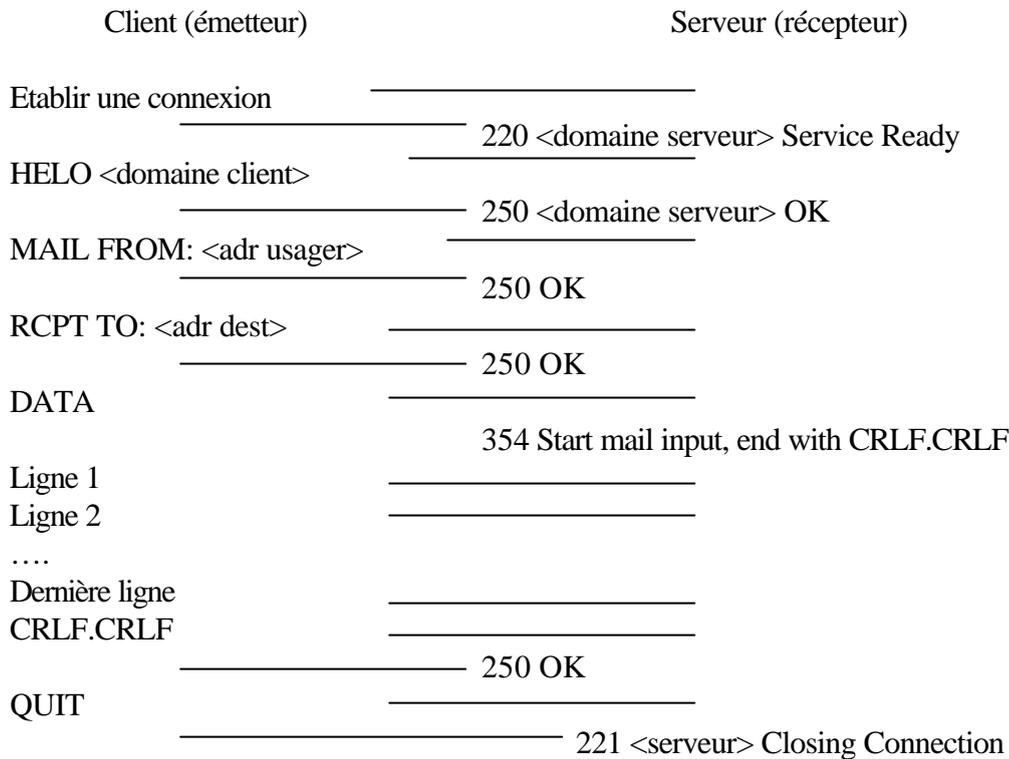
I.6) Quelles sont les différentes étapes de la transmission d'un courrier électronique dans le cas habituel d'un prestataire de service d'accès Internet ou d'une entreprise de taille significative. Précisez les matériels (ordinateurs), les logiciels (agent utilisateur, serveur de messagerie, donnez des 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'utilisateur 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 crée 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](mailto:smtp.cnam.fr)

POP server [pop.cnam.fr](mailto:pop.cnam.fr)

Mail account: natkin

Mail adress: [natkin@cnam.fr](mailto:natkin@cnam.fr)

Reply adress: [natkin@cnam.fr](mailto:natkin@cnam.fr)

Le protocole POP fonctionne en mode connecté avec TCP. Après avoir fourni un message d'accueil 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 ::= "+OK" [" "]*text] CRLF
```

```
    [multiline CRLF "." CRLF]
```

```
statresponse ::= "+OK" [" "1*DIGIT" "1*DIGIT]
```

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

multiline::=1\*<n'importe quelle chaîne caractère ASCII ne comprenant pas la chaîne CRLF>."CRLF">  
 mailbox::=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

0M8R4KGxGuEAAAAAAAAAAAAAAAAAAAAAAAAAPgADAP7/CQAGAAAAAAAAAAAAAAAAAB
AAAAARQAAAA
AAAAEAAARwAAAAEAAAD+////AAAAEQAAAD////////////////////////////////////
////////////////////////////////////
```

```
--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).

## Exercice 16 : DNS et messagerie Internet/l'approche SPF

Dans ce problème nous étudions deux méthodes de lutte contre le courrier non sollicité ('spam'), qui sont basées sur les relations entre la messagerie SMTP et le service de nom de domaines DNS. De nombreuses autres méthodes ont été proposées que nous n'étudions pas ici, en particulier des méthodes de filtrage visant à sélectionner les courriers sur le contenu.

Dans ce texte, une entreprise connectée à l'Internet possède un nom de domaine (comme par exemple fai.com..). Cette entreprise gère un service de courrier électronique au moyen de différents serveurs de messagerie SMTP. Un hôte de l'Internet, dont l'adresse IP est par exemple a.b.c.d et dont le nom de domaine DNS est par exemple hote.domaine.. , utilise le protocole SMTP pour envoyer un courrier électronique à un destinataire d'adresse usager@fai.com.

1) Dans le DNS rappelez la définition de la notion d'enregistrement ressource (RR Resource record) ?

2) A quoi servent les types d'enregistrements A et MX ?

3) Quelles sont les actions que doit réaliser un client SMTP (qui tourne par exemple ici sur hote.domaine..) pour localiser le serveur de courrier (par exemple de usager@fai.com..) avant de commencer les échanges SMTP (actions DNS et interface transport)?

Pour éviter une partie des courriers non sollicités ('spam') certaines entreprises, pour effectuer une première vérification simple, imposent les deux règles suivantes.

a) Tout hôte de l'Internet qui expédie des courriers électroniques à un serveur de courrier doit avoir obligatoirement un enregistrement ressource de type PTR dans sa base de donnée DNS permettant d'effectuer une recherche inverse.

b) Le nom de domaine d'un hôte émetteur de courrier doit posséder un enregistrement ressource de type A.

4) Comment l'enregistrement ressource de type PTR est il utilisé dans le DNS pour réaliser une recherche inverse?

5) En utilisant uniquement les informations échangées dans le cadre des protocoles SMTP et TCP, quelles sont les informations dont dispose un serveur SMTP pour identifier l'émetteur d'un courrier électronique qu'il vient de recevoir (trois informations) ?

6) Quelle sont les vérifications que veut effectuer un serveur SMTP d'une entreprise comme fai.com qui impose aux émetteurs de courriers les deux règles a) et b) (quelles sont les requêtes DNS utilisées) ?

7) L'administrateur d'un site émetteur de courrier, souhaite vérifier que son DNS est correctement configuré pour que ses courriers ne soient pas rejetés par une entreprise comme fai.com.. qui applique les règles a) et b). Il utilise l'outil nslookup disponible sous Unix ou Windows.

Quelles commandes doit-il taper pour vérifier qu'il satisfait les deux règles a) et b) (expliquez la formation des noms de domaines utilisés pour les lignes de commande) ?

Pour exprimer votre réponse vous pouvez utiliser aussi dig ou host si vous les connaissez mieux.

Cependant les deux règles a) et b) n'évitent plus beaucoup de courriers non sollicités car les entreprises qui font du spam le font, non pas à partir de machines dans leur domaine, mais à partir de machines appartenant à d'autres domaines de l'Internet qui ont été infestées par des virus chargés d'émettre les courriers indésirables. Une proposition de vérification qui améliore les règles a) et b) a été récemment acceptée par l'IETF. L'ensemble des nouveaux mécanismes est baptisé SPF pour 'Sender Policy Framework'. La nouvelle vérification est basée sur la création dans le DNS d'un nouveau type d'enregistrement ressource qui contient pour un domaine les machines habilitées à émettre du courrier.

8) Le nouvel enregistrement ressource a d'abord été défini en utilisant le type existant TXT. Pourquoi avoir utilisé ce type existant ?

Il existe de nombreuses directives en SPF pour spécifier une machine habilitée à émettre du courrier. Une façon très simple d'appliquer une politique SPF, consiste à créer dans le DNS l'enregistrement suivant :

```
domaine. IN TXT "v=spf1 mx -all"
```

domaine. : le nom de domaine concerné par l'enregistrement ressource

IN TXT : enregistrement ressource type TXT pour le réseau Internet

"v=spf1" : une valeur obligatoire pour indiquer une utilisation du RR TXT en SPF version 1

"mx" : spécifie que tous les serveurs de courrier du domaine sont des émetteurs autorisés

"-all" : indique que tous les courriers qui ne satisfont pas la contrainte sont rejetés

9) Quelles sont les vérifications qui sont effectuées par un serveur SMTP d'une entreprise qui applique la politique SPF vis-à-vis d'un émetteur ayant inséré dans son DNS l'enregistrement ressource précédent (quelles sont les requêtes DNS utilisées) ?

Il existe différentes autres variantes permettant de spécifier des émetteurs de courrier autorisés que nous n'étudions pas ici: directement par le nom de domaine d'un hôte ou par une plage d'adresses IP autorisées etc ...

10) Supposons que la politique SPF soit généralisée, c'est-à-dire qu'on applique des vérifications d'autorisation d'émission, alors le volume des courriers non sollicités pourrait être mieux contrôlé. Quels mécanismes additionnels les fournisseurs d'accès Internet peuvent-ils mettre en oeuvre (selon quels critères) pour effectuer ce contrôle ?

## Exercice 17 : Messagerie Internet: Analyse du contenu d'un courrier

La messagerie Internet a délivré un courrier de publicité commerciale. Vous souhaitez analyser différents aspects concernant ce courrier au moyen des informations qu'il contient pour rechercher d'éventuels problèmes. Vous disposez d'un client de messagerie qui vous permet d'afficher le détail des informations contenues dans ce courrier et vous obtenez la liste suivante:

```
Return-Path: <infos@topachat.com>
X-Original-To: gerard@cnam.fr
Delivered-To: gerard@cnam.fr
Received: from brangien.cnam.fr (brangien.cnam.fr [163.173.128.20])
  by copernic.cnam.fr (Postfix) with ESMTTP id D0E5DEB172
  for <gerard@cnam.fr>; Thu, 8 Mar 2007 06:34:09 +0100 (CET)
Received: from localhost (localhost [127.0.0.1])
  by brangien.cnam.fr (Postfix) with ESMTTP id C46AD12390B
  for <gerard@cnam.fr>; Thu, 8 Mar 2007 06:34:09 +0100 (CET)
X-Virus-Scanned: amavisd-new at cnam.fr
Received: from brangien.cnam.fr ([127.0.0.1])
  by localhost (brangien.cnam.fr [127.0.0.1]) (amavisd-new, port 10024)
  with ESMTTP id iJMgWyOys9h3 for <gerard@cnam.fr>;
  Thu, 8 Mar 2007 06:34:08 +0100 (CET)
Received: from mta1.topachat-clust.com (mta1.topachat-clust.com
[195.167.228.242])
  by brangien.cnam.fr (Postfix) with ESMTTP id 3584B123AD0
  for <gerard@cnam.fr>; Thu, 8 Mar 2007 06:34:06 +0100 (CET)
Received: from topachat.com (localhost [127.0.0.1])
  by mta1.topachat-clust.com (8.12.9+Sun/8.12.9) with ESMTTP id
1285Y1Vg001884
  for <gerard@cnam.fr>; Thu, 8 Mar 2007 06:34:05 +0100 (MET)
Content-Transfer-Encoding: binary
Content-Type: multipart/alternative; boundary="=_la-cesure-
=_1173332045212331209"
MIME-Version: 1.0
Organization: infos@topachat.com
Reply-To: infos@topachat.com
X-Comment: CODEOP=20070307-1-39,QOS=1,ID=Z2VyYXJkQGNuYW0uZnI=
X-Mailer: mailing.1.8.0 by FG
Date: Thu, 8 Mar 2007 05:34:05 UT
From: infos@topachat.com
To: gerard@cnam.fr
Subject: =?ISO-8859-1?Q?Le compte =E0 rebours a commenc=E9 ...?=
Message-Id: <1173332045.Z2VyYXJkQGNuYW0uZnI=.1.20070307-1-39@topachat.com>
```

This is a multi-part message in MIME format.

```
--_la-cesure--_1173332045212331209
Content-Disposition: inline
Content-Length: 198
Content-Transfer-Encoding: 8bit
Content-Type: text/plain; charset="iso-8859-1"
MIME-Version: 1.0
Date: Thu, 8 Mar 2007 05:34:05 UT
```

Bonjour,

La newsletter au format Html se trouve ici:

http://www.topachat.com/mail/07032007.htm  
L'Anim'Team  
infos@topachat.com

Pour vous désinscrire:

http://www.topachat.com/pages/black.php

--\_la-cesure==\_1173332045212331209

**Content-Disposition:** inline

**Content-Length:** 10830

**Content-Transfer-Encoding:** 8bit

**Content-Type:** text/html; charset="iso-8859-1"

**MIME-Version:** 1.0

**Date:** Thu, 8 Mar 2007 05:34:05 UT

<html>

<head>

<title>Le compte &agrave; rebours &agrave; commencer ...</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<style> deco { background-repeat: no-repeat; }

v12 { font-size: 14px; }

</style>

</head>

<body bgcolor="#FFFFFF" text="#000000"

background="http://promo.topachat.com/mail/images/fond2px.gif">

<center>

<p><font face="arial" style="font-size: 9px; color: #000000" >Si ce message  
ne s'affiche pas correctement, suivez ce lien: <a

href="http://www.topachat.com/mail/07032007.htm">http://www.topachat.com/ma  
il/07032007.htm</a><br>

----- Ici se poursuit un document HTML beaucoup trop long pour être  
conservé. -----

----- Le courrier électronique se termine par les lignes  
suivantes : -----

</body>

</html>

--\_la-cesure==\_1173332045212331209--

1) Le courrier précédent est défini à la base par les spécifications de la RFC 822. Rappelez les grandes lignes de cette spécification (rappelez les principes généraux qui gouvernent la structure des messages dans la messagerie Internet). Vous illustrerez votre réponse en prenant pour chaque point cité un exemple dans le contenu du courrier précédent.

2) Ce message est également gouverné par les spécifications de l'extension MIME. Rappelez les grandes lignes de cette extension. Donnez des exemples significatifs de l'incidence des extensions apportées par le format MIME sur le courrier précédent.

3) On trouve au début du courrier plusieurs lignes d'informations commençant par 'received'. Prenons par exemple la première :

**Received:** from brangien.cnam.fr (brangien.cnam.fr [163.173.128.20])

by copernic.cnam.fr (Postfix) with ESMTP id D0E5DEB172

for <gerard@cnam.fr>; Thu, 8 Mar 2007 06:34:09 +0100 (CET)

A quoi sert cette information ? Commentez les différents champs dans cette directive ?

4) Pouvez vous retracer l'histoire de l'acheminement de ce courrier (lieux par lesquels il est passé et heures) ? Utilisez cette trace pour expliquer le mieux possible les différents traitements que ce courrier a subi.

5) Expliquez la directive suivante.

**Subject:** =?ISO-8859-1?Q?Le compte =E0 rebours a commenc=E9 ...?=  
Indications : Quel est son objectif ? Pourquoi cette forme de présentation bizarre ?

6) Expliquez la ligne d'entête suivante :

**Content-Type:** multipart/alternative; boundary="\_-la-cesure-=\_1173332045212331209"

7) Finalement, ce courrier vous parait-il présenter un problème ou est-il normal (son émetteur est-il connu, le message a t'il été relayé un nombre anormal de fois, son contenu est-il suspect) ?

## Exercice 18 : 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'orthographe 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 nslookup 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 ?

## Exercice 19 : IMAP 'Internet Message Access Protocol'

Le dialogue suivant est un test du protocole IMAP ('Internet Message Access Protocol' en version 4 RFC1730 IMAP4 de décembre 1994).

De manière à faciliter la lecture, les ensembles cohérents (une commande suivie d'une réponse) ont été espacés par des lignes blanches.

En IMAP4, chaque commande est préfixée par un identificateur alphanumérique (une étiquette ou 'tag') qui est reprise dans la réponse. Normalement une étiquette différente doit être utilisées pour chaque commande de manière à associer commandes et réponses. Ici nous avons utilisé à chaque fois les mêmes deux caractères ' .' (caractère point suivi de caractère espace).

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

```
C:\>telnet imap.cnam.fr.. 143
Trying 163.173.128.12...
Connected to imap.cnam.fr.
Escape character is '^]'.
* OK [CAPABILITY IMAP4rev1 UIDPLUS CHILDREN NAMESPACE THREAD=ORDEREDSUBJECT
THREAD=REFERENCES SORT QUOTA IDLE] Courier-IMAP ready. Copyright 1998-2005
Double Precision, Inc. See COPYING for distribution informaotion.

. LOGIN gerard le_mot_de_passe_secret_a_été_donné_ici
. OK LOGIN Ok.

. SELECT INBOX
* FLAGS (\Draft \Answered \Flagged \Deleted \Seen \Recent)
* OK [PERMANENTFLAGS (* \Draft \Answered \Flagged \Deleted \Seen)] Limited
* 24 EXISTS
* 24 RECENT
* OK [UIDVALIDITY 1025189503] Ok
* OK [MYRIGHTS "acdilrsw"] ACL
. OK [READ-WRITE] Ok

. SEARCH since 1-july-2007
* SEARCH
. OK SEARCH done.

. FETCH 1 (BODY.PEEK[HEADER.FIELDS (SUBJECT)])
* 1 FETCH (BODY[HEADER.FIELDS ("SUBJECT")] {25}
Subject: Re: Valeur C
)
. OK FETCH completed.

. FETCH 6 (BODY.PEEK[HEADER.FIELDS (SUBJECT)])
* 6 FETCH (BODY[HEADER.FIELDS ("SUBJECT")] {33}
Subject: GRIDtoday Daily News
)
. OK FETCH completed.

. LOGOUT
* BYE Courier-IMAP server shutting down
. OK LOGOUT completed
```

Perte de la connexion à l'hôte.

Appuyez sur une touche pour continuer...

1) Pourquoi peut-on simuler un client IMAP en utilisant la commande suivante ?

```
C:\>telnet imap.cnam.fr.. 143
```

2) Pourquoi en IMAP doit-on s'authentifier ?

```
. LOGIN gerard le_mot_de_passe_secret_a_été_donné_ici  
. OK LOGIN Ok.
```

3) Pourquoi doit-on obligatoirement sélectionner un dossier de courrier (une boîte à lettre) pour travailler en IMAP

```
. SELECT INBOX
```

4) La commande FETCH permet comme en POP3, de récupérer un courrier selon un filtrage perfectionné. Qu'est ce qui a été récupéré ici ? Pourquoi IMAP est-il présenté comme économisant la bande passante du réseau ?

```
. FETCH 1 (BODY.PEEK[HEADER.FIELDS (SUBJECT)])
```

5) Il y a longtemps on a pu réaliser la relève du courrier en utilisant le protocole SMTP. On voit que IMAP est dérivé de SMTP mais il en est maintenant très éloigné. Pourquoi (quel avantage a-t-on en utilisant IMAP) ?

## Exercice 20 : WAP ('Wireless Application Protocol')

Le WAP ('Wireless Application Protocol') a pour objectif de permettre l'accès à des ressources de l'Internet (informations ou services) au moyen de clients mobiles légers qui sont essentiellement des téléphones portables ou des organiseurs (PDA 'Portable Document Assistant'). Les applications visées sont voisines de celles du Web en diminuant les fonctionnalités attendues : recherches simples (numéros de téléphone, adresses, horaires de transports...), accès à des informations courtes (conditions météo, cours de bourse, trafic routier ...), transactions simples (achat de billets, ...).

Le WAP est défini par un ensemble de normes élaborées par le Wapforum (regroupement de constructeurs comme Ericsson, Motorola, Nokia, ...) à partir de 1997. Le WAP dérive des standards Internet du Web (TCP/IP, HTML, XML, HTTP) en les adaptant à l'univers des téléphones portables c'est à dire à de petits terminaux sans fils. En particulier le WAP définit la notion de micro navigateur ('micro browser') qui minimise la demande en ressources matérielles (mémoire, puissance processeur, débit de communication) sur un poste client.

### Première partie : le langage WML ('Wireless Markup Language')

Un micro navigateur WAP peut afficher des informations des lors qu'elles sont spécifiées dans le langage de balise WML pour 'Wireless Markup Language'. WML est un langage de balises qui réalise une partie des fonctionnalités offertes par HTML. Les ensembles d'informations échangées en WML sont appelés des 'decks'. Un deck est défini par la balise wml. Les decks sont composés d'unités d'affichage baptisées cartes (définies par la balise 'card'). Sur un poste client on affiche une carte à la fois. Une carte peut contenir du texte, des liens (en particulier sur d'autres cartes), des images. Par exemple un texte affiché est encadré par la balise p (pour paragraphe). Quand un deck WML est accédé, l'ensemble des cartes qu'il contient est chargé à partir du serveur WAP. Une navigation entre cartes est alors possible, sous le contrôle du micro navigateur local, sans aucun accès au serveur distant.

Voici un exemple très simple de document wml (un deck avec une carte). Dans cet exemple on voit que WML est défini comme un dialecte XML version 1.0 (un document WML est tout d'abord un document XML 1.0).

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>

<card id="Bourse" title="A la bourse de Paris">
<p>
A l'ouverture ce matin le CAC40 ..
</p>
</card>

</wml>
```

Le résultat affiché sur un micro navigateur pour le source précédent pourrait ressembler à :

```
-- A la bourse de Paris --  
  
A l'ouverture ce matin le  
CAC 40 ..
```

La forme graphique précédente ne correspond pas à un micro navigateur réel et est donnée à titre purement indicatif pour comprendre l'exemple.

1) A quoi correspondent les trois premières lignes du document ?

```
<?xml version="1.0"?>  
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"  
"http://www.wapforum.org/DTD/wml_1.1.xml">
```

2) Le document XML précédent est-il bien formé ?

3) Que devrait-on faire pour vérifier que ce document est valide ?

Le second exemple suivant montre un document wml un peu plus compliqué avec un deck et deux cartes (carte1 et carte2) qui s'enchaînent. La première carte permet de sélectionner une seule réponse parmi un ensemble de réponses possibles au moyen de boutons. Le résultat est passé dans une variable code\_reponse à la seconde carte qui affiche la sélection réalisée.

```
<?xml version="1.0"?>  
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"  
"http://www.wapforum.org/DTD/wml_1.1.xml">  
  
<wml>  
  
<card id="carte1" title="Menu">  
<do type="accept" label="Choisissez">  
  <go href="#carte2"/>  
</do>  
<p>  
<select name="code_reponse">  
  <option value="Meteo"> Infos Meteo </option>  
  <option value="Trafic"> Infos Trafic </option>  
  <option value="Bourse"> Infos Bourse </option>  
</select>  
</p>  
</card>  
  
<card id="carte2" title="Choix">  
<p>  
Vous avez selectionne : $(code_reponse)  
</p>  
</card>  
  
</wml>
```

La première carte affichée, après avoir sélectionné Infos météo, pourrait être :

```
----- Menu -----  
Infos Meteo     
Infos Trafic    
Infos Bourse    
Choisissez
```

La seconde carte affichée pourrait être :

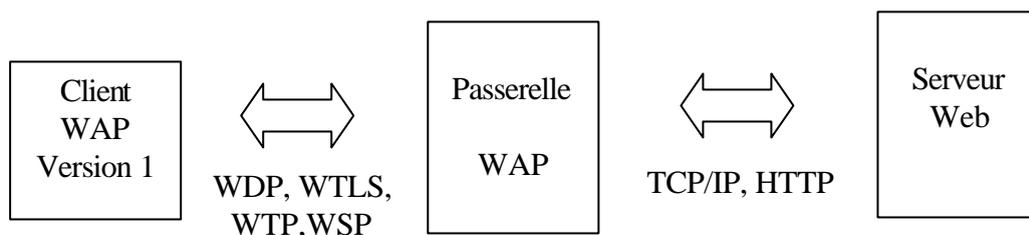
```
----- Choix -----  
Vous avez selectionne : Meteo
```

- 4) Quels sont les éléments et les attributs présents dans l'exemple précédent ? Ce document possède-t-il des éléments vides ?
- 5) Quel est l'arbre résultant de l'analyse syntaxique du document précédent ?
- 6) Quels sont les principaux avantages du choix de XML pour définir WML ? La discussion ne doit pas consister à recopier tous les avantages de XML mais à discuter des avantages d'une solution XML par rapport à une solution de type HTML, dans le cas précis du WAP. Par exemple on aurait pu choisir une version simplifiée de HTML qui aurait limité dans HTML les balises utilisables pour des terminaux simples.

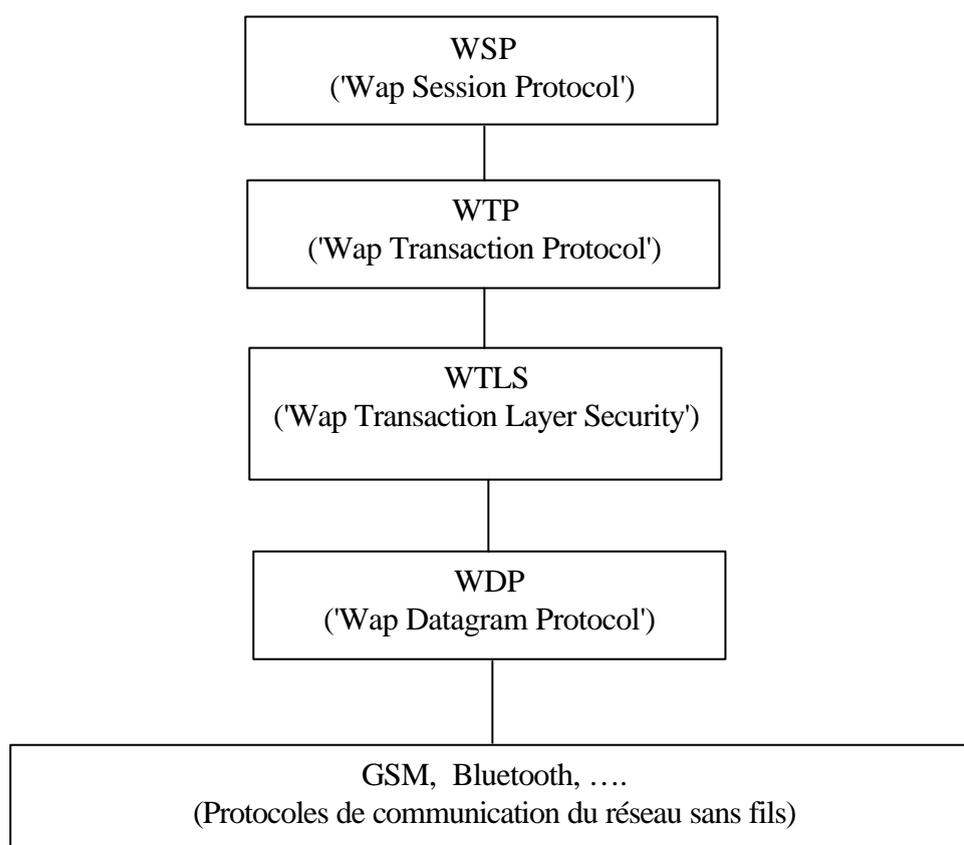
## Seconde partie : les protocoles de communication du WAP

La suite des protocoles de communication du WAP a pour objectif de réaliser des fonctionnalités similaires à celles de la suite des protocoles Internet. Cette suite a été définie en deux étapes. Dans les versions 1 (1.0, 1.1, etc.) les protocoles (voir plus loin WDP, WTLS, WTP, WSP) sont complètement spécifiques des appareils aux possibilités limitées que sont les téléphones portables ou les PDA. Ils sont incompatibles avec les protocoles de la suite Internet/Web. À partir de la version 2 le standard WAP inclut, en plus des protocoles de la version 1, la suite des protocoles standards de l'Internet et du web (WWW World Wide Web).

Pour dialoguer avec un serveur Web habituel, les communications WAP en version 1 doivent donc tout d'abord être réalisées entre un appareil WAP version 1 et une passerelle qui transforme les messages et les formats spécifiques WAP version 1 en messages et formats habituels de l'Internet et du Web.



Dans la version 1 les protocoles du WAP comportent quatre couches principales qui utilisent les services d'un protocole de communication de réseau sans fil:



7) WDP ('Wap Datagram Protocol') est un protocole qui réalise au dessus du protocole de communication du réseau sans fil (par exemple le protocole GSM des téléphones portables) un mode de communication très analogue à celui du protocole UDP. En fonction des caractéristiques de la couche UDP que vous connaissez quel est l'objectif poursuivi par la couche WDP dans la pile des protocoles WAP ?

8) WTLS est un protocole optionnel qui réalise dans l'univers WAP des fonctions similaires au protocole Internet SSL/TLS. Selon les principales fonctions réalisées par SSL/TLS que vous rappellerez pourquoi avoir introduit cette couche dans la pile des protocoles WAP en version 1.1 ?

9) WTP ('Wap Transaction Protocol') n'est pas un protocole transactionnel au sens de la gestion de l'accès concurrent aux données et de la validation atomique. WTP dans le mode 1 de

fonctionnement réalise essentiellement un contrôle d'erreur par acquittement positif et temporisateur sur les messages acheminé par WDP. Dans le mode 2 , WTP implante un mode client serveur à trois messages émission, acquittement positif, réponse. Par ailleurs WTP effectue également la fragmentation et le réassemblage des messages longs. Quelle est dans le monde Internet/Web le protocole de transport utilisé par le protocole HTTP? Pourquoi avoir dans le WAP introduit le protocole WTP avec les fonctionnalités précédentes succinctement décrites de contrôle d'erreur et de fragmentation ?

10) WSP ('Wap Session Protocol') possède deux modalités: avec et sans connexion. Dans sa modalité WSP/B , WSP est un protocole sans connexion, sans état qui réalise un sous ensemble du protocole HTTP 1.1 . Il comporte pour cela des messages de requête et de réponse et achemine des informations similaires à HTTP 1.1 (ligne initiale, entêtes, corps de message). Une différence essentielle de WSP avec HTTP 1.1 est que les différents champs caractéristiques d'une requête ou d'une réponse WSP sont codés en binaire alors qu'ils sont en format texte en HTTP. De même, il a été défini un format WML binaire baptisé WBXML ('Wap Binary XML') qui encode en binaire au moyen d'un ensemble de règles d'encodage les documents WML échangés dans les messages WSP. Un document WML encodé en WBXML possède un format standard avec une entête, un dictionnaire de chaînes de caractères avec leur encodage et un ensemble d'éléments encodés. Pourquoi avoir défini WSP/B comme un sous ensemble de HTTP 1.1 codé en binaire de même qu'un codage binaire des documents WML ?

## Exercice 21 : Services sur la toile ('Web Services')

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/finacial">
        <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](mailto: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 permettre 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 où 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).

## Exercice 22 : Protocole SIP ('Session Initiation Protocol')

Le protocole SIP ('Session Initiation Protocol') a été conçu à partir de 1996 par le groupe de travail de l'IETF MMUSIC (Multiparty MULTimedia Session Control), une émanation du groupe de travail sur la diffusion dans l'Internet Mbone (Multicast backBONE). Il a ensuite été développé jusqu'à ce jour.

SIP est destiné à la création, la modification et la terminaison de sessions d'échanges de données multimédias entre usagers du réseau Internet. Ses domaines d'applications principaux sont donc ceux de l'ouverture et de la fermeture des sessions d'échanges de textes (messagerie instantanée), audio (communications téléphoniques), audio/vidéo (conférences multimédia, diffusion de films, jeux vidéos en réseau). De manière générale tout échange de données multimédia temps réel en pair à pair peut être supervisé en utilisant SIP. SIP est assez largement diffusé depuis 2001 surtout dans le domaine de la téléphonie. La RFC initiale est de 1999 et les principales RFC actuelles sont les RFC 3261 à 3266 parues en 2002.

SIP est un protocole client serveur de niveau application. Il sert pour la signalisation c'est-à-dire la mise en relation d'entités distantes et l'échange d'informations définissant le contexte d'une communication multimédia (protocole de transport utilisé, codage utilisé, ...). SIP doit être clairement distingué de la transmission proprement dite des données multimédia (paquets d'échantillons de son, d'images ou de données informatiques). On utilise généralement alors le protocole de transport de données multimédia temps réel RTP/RTCP ('Real-time Transport Protocol' et son contrôle 'Real-time Control Protocol') et/ou tous les autres protocoles associés aux communications avec qualité de service RTSP ('Real-time Streaming Protocol'), RSVP ('Resource Reservation Protocol') ....

Pour faire de la signalisation, SIP réutilise et étend de très nombreux aspects existants dans des protocoles Internet : HTTP (les formats de messages), DNS (la gestion des noms et adresses), SMTP (adressage, entêtes et codage MIME des données multimédia). SIP utilise aussi un protocole moins connu SDP ('Session Description Protocol') spécialisé dans la négociation de paramètres d'échanges de données multimédia. SIP peut fonctionner soit sur TCP soit sur UDP (on peut utiliser UDP pour des raisons d'efficacité).

1) Rappelez la différence entre le fonctionnement des communications en mode connecté et en mode non connecté ?

2) L'établissement des connexions au niveau application a été vu par exemple dans TELNET ou dans SMTP. C'est aussi un problème majeur dans une communication multimédia. Pourquoi ? Donnez le maximum de raisons (de tâches à accomplir) qui justifient un protocole particulier de connexion pour des communications multimédia interactives ?

### SIP et DNS

Pour réaliser des opérations SIP deux utilisateurs ('user agent') dialoguent en échangeant des messages selon un protocole client serveur de sorte que chaque utilisateur possède une partie client UAC (User Agent Client) et une partie UAS (User Agent Server). Par ailleurs, pour arriver à se mettre en relation, les utilisateurs SIP utilisent des serveurs SIP. Un serveur SIP peut jouer différents rôles :

a) Mandataire ('proxy server'). C'est un relais des messages de signalisation SIP qui réalise des fonctions associées à un proxy. Par exemple pour trouver le destinataire effectif d'un message SIP, un serveur proxy SIP utilise un serveur d'annuaires (serveur de localisation 'location server').

b) Redirection des appels ('redirect server'). C'est un serveur qui est capable de rediriger des appels vers des serveurs mieux appropriés pour rendre le service (par exemple après le déplacement du correspondant dans le réseau Internet).

c) Enregistrement d'une adresse ('registrar server'). Ce type de serveur accepte des requêtes d'enregistrement de localisation de la part des usagers (correspondance nom logique adresse IP). Il utilise pour cela une gestion d'annuaires de correspondants ('location server').

Sauf s'il est capable d'établir seul une communication avec un correspondant car il a toutes les informations nécessaires, un utilisateur SIP doit en général tout d'abord trouver un serveur SIP dans un domaine pour réaliser les fonctions qui viennent d'être décrites. Un utilisateur utilise pour cela le DNS. L'enregistrement ressource du DNS utilisé ici est de type SRV qui est un raccourci pour serveur. SRV sert à définir, pour un domaine, le nom d'un hôte qui rend un service donné. Un tel outil peut concerner tous les services offerts dans l'internet. Ceux qui correspondent à des ports bien connus comme http, ftp, ... mais aussi des services commerciaux propriétaires n'ayant pas de port bien connu ou même des services non ouverts définis en interne dans une entreprise. La structure de l'enregistrement SRV est la suivante :

srvce.prot.name ttl class rr pri weight port target

srvce : définit le type du service. Le nom du service doit être précédé selon la dernière RFC d'un soulignement \_ (le caractère underscore sert à distinguer les services des autres noms de domaine). Il existe une liste de services autorisés (http, ftp, ldap, ...).

prot : le nom du protocole de transport utilisé précédé de soulignement \_ (TCP , UDP, ...).

name : le nom du domaine dans lequel on veut trouver le serveur (par exemple cnam.fr).

pri : la priorité du serveur (un entier positif à partir de 0, les priorités les plus élevées sont les plus petites valeurs).

weight : la capacité du serveur pour la répartition de charge (par exemple si l'on a deux serveurs avec valeurs 1 et 4 le second serveur est quatre fois plus puissant que le premier).

port : le numéro du port permettant d'accéder au service

target : le nom de domaine du serveur (de l'hôte qui rend le service). Un point indique que le service en question n'est pas rendu pour ce domaine.

Un exemple de définition d'un service pourrait être au CNAM :

```
_http_tcp.cnam.fr. 86400 IN SRV 0 5 80 www.cnam.fr.
```

3) Dans le DNS, sans utiliser SRV, on peut obtenir des informations de localisation de serveurs. Pouvez vous citer deux enregistrements ressources utilisables à cette fin?

4) Sachant que le port imap est le 143, quel enregistrement ressource devrait définir le CNAM pour indiquer à ses usagers que son serveur imap s'appelle imap.cnam.fr (1 point) ? Quel enregistrement ressource devrait être défini pour indiquer qu'il n'y a pas de serveur pop3 (si le CNAM n'avait pas de serveur pop) ?

5) Voici par exemple, ce qui concerne SIP dans le fichier de zone du serveur DNS du département informatique de l'université de Columbia (cs.columbia.edu):

```
_sip_tcp.cs.columbia.edu SRV 20 0 5060 backbay.cs.columbia.edu  
_sip_tcp.cs.columbia.edu SRV 0 0 5060 conductor.cs.columbia.edu  
_sip_tcp.cs.columbia.edu SRV 10 0 5060 erlang.cs.columbia.edu  
_sip_udp.dynamicsoft.com SRV 0 0 5060 dyn-tx-app-006.dfw.dynamicsoft.com
```

Une application SIP interroge le DNS sur les serveurs d'annuaires SIP en TCP de cs.columbia.edu. Elle obtient trois réponses. Quel serveur doit normalement être choisi en premier (justifiez votre réponse) ?

6) Dans quels cas et selon quelles stratégies doit-on utiliser un autre serveur ?

7) Est-ce que le serveur d'une application pour un domaine se trouve dans ce domaine ?

## Adressage SIP

Le schéma d'adressage associé au protocole SIP permet d'attribuer à tout utilisateur SIP dans l'Internet un nom global. Dans la norme SIP ce nom d'utilisateur final est défini dans le cadre des URL comme l'ensemble des champs suivants :

`sip:user:password@host:port;uri-parameters?headers`

Ce qui donne comme exemples :

`sip:jean:secret@hote.cnam.fr:9950;transport=tcp?subject=RV&priority=urgent`

`sip:+1-212-555-1212:1234@gateway.com;user=phone`

**User:** c'est l'identificateur d'un usager sur l'hôte adressé. Il s'agit d'une chaîne de caractère quelconque. Elle peut aussi être définie dans le cadre de l'une des applications utilisatrices de SIP : un numéro de téléphone selon la numérotation habituelle, un identificateur dans un logiciel de messagerie instantanée (chat)... User peut être omis si l'hôte n'héberge qu'un unique destinataire (exemple un téléphone IP).

**Password :** un mot de passe en clair (optionnel).

**Host :** un nom DNS absolu (FQDN) ou une adresse IPV4 ou IPV6.

**Port :** un numéro de port (le port par défaut SIP en TCP ou UDP est le port 5060).

**Uri-parameters :** une liste de couples nom\_paramètre=valeur\_parametre séparés par des point virgules qui précisent l'interprétation de l'URI (on voit comme exemple le transport utilisé, la nature de la zone usager qui peut être un numéro de téléphone, ...).

**Headers :** une liste de couples type\_entete=valeur séparés par des & (et commerciaux). Ces définitions sont destinées à être transformées en des lignes d'entête dans la requête SIP (analogue d'une requête http). Voir par exemple subject=rendezvous qui est transformé en une ligne d'entête Subject : rendezvous

8) La structure générale d'une URL dans le schéma SIP reprend certains aspects d'une adresse de courrier électronique Internet SMTP (dans les URL le schéma des adresses électroniques s'appelle MAILTO). Quels sont les aspects hérités de l'adressage du courrier électronique (MAILTO).

9) La structure générale d'une adresse dans le schéma SIP hérite certains aspects des adresses de la toile mondiale WWW (schéma HTTP). Quels sont les aspects hérités de l'adressage de la toile (HTTP)?

10) La partie Host permet de définir une adresse IP, ou un nom de domaine DNS. On peut alors définir un hôte précis, un service ou un domaine de niveau intermédiaire dans la hiérarchie de désignation du DNS. Quel usage peut-on faire de ces différents moyens de désignation ?

## Protocole SIP

De manière générale le protocole SIP est visiblement dérivé du protocole HTTP. Mais, comme ses objectifs sont différents, il présente en fait des différences significatives. En particulier, il implante des méthodes différentes dont les principales sont les suivantes :

**INVITE** : initie des sessions dont la description est incluse dans le corps.

**ACK** : est une réponse à l'invite et confirme l'établissement d'une session.

**BYE** : termine une session

**CANCEL** : arrête un dialogue d'ouverture de session en cours.

**OPTIONS** : permet de définir les fonctions réalisées par une implantation.

**REGISTER** : relie une adresse SIP à une localisation (adresse IP).

Un échange de base caractéristique en SIP est le suivant (exemple tiré de D.Sisalem et J. Kutan 'Understanding SIP'). Il s'agit d'une ouverture de session SIP réussie entre deux usagers distants (un agent client UAC et un agent serveur UAS) :

```
1 INVITE sip:UserB@there.com SIP/2.0
2 Via: SIP/2.0/UDP here.com:5060
3 From: BigGuy<sip:UserA@here.com>
4 To: LittleGuy<sip:UserB@there.com>
5 Call-ID: 12345600@here.com
6 CSeq: 1 INVITE
7 Subject: Happy Christmas
8 Contact: BigGuy<sip:UserA@here.com>
9 Content-Type: application/sdp
10 Content-Length: 147
11
12 v=0
13 o=UserA 2890844526 2890844526 IN IP4 here.com
14 s=Session SDP
15 c=IN IP4 100.101.102.103
16 t=0 0
17 m=audio 49172 RTP/AVP 0
18 a=rtpmap: 0 PCMU/8000
```

La réponse est la suivante (comme en HTTP on peut aussi avoir des réponses intermédiaires comme par exemple '100 Trying' ou '180 Ringing') :

```
1 SIP/2.0 200 OK
2 Via: SIP/2.0/UDP here.com:5060
3 From: BigGuy<sip:UserA@here.com>
4 To: LittleGuy<sip:UserB@there.com>; tag=65a35
5 Call-ID: 12345601@here.com
6 CSeq: 1 INVITE
7 Subject: Happy Christmas
8 Contact: LittleGuy <sip:UserB@there.com>
9 Content-Type: application/sdp
10 Content-Length: 134
11
12 v=0
13 o=UserB 2890844527 2890844527 IN IP4 there.com
14 s=Session SDP
15 c=IN IP4 110.111.112.113
16 t=0 0
17 m=audio 3456 RTP/AVP 0
```

18 a=rtpmap:0 PCMU/8000

Le message qui termine l'échange d'ouverture de session est un acquittement positif :

```
1 ACK sip:UserB@there.com SIP/2.0
2 Via: SIP/2.0/UDP here.com:5060
3 From: BigGuy<sip:UserA@here.com>
4 To: LittleGuy<sip:UserB@there.com>
5 Call-ID: 12345600@here.com
6 CSeq: 1 ACK
7 Content-Length: 0
8
```

11) Dans un document sur SIP, on parle à propos de la requête INVITE d'idempotence. Rappelez la définition de l'idempotence.

12) La réaction d'un utilisateur distant à une demande d'ouverture de connexion peut-être longue. Pourquoi l'INVITE doit plutôt être implantée en étant idempotente ?

13) Qu'est ce qui pourrait faire que l'invite ne soit pas considérée comme idempotente ?

14) En utilisant vos connaissances des protocoles SMTP et HTTP, commentez les lignes 1, 3, 4, 5, 7, 9, 10, 11 de la requête INVITE. Que pouvez vous interpréter des lignes 12 à 18 de la requête INVITE?

15) L'entête VIA (dans l'INVITE en ligne 2 Via: SIP/2.0/UDP here.com:5060) indique le chemin que doit emprunter la requête (un relais qui doit être visité par la requête). Quel est le mécanisme analogue dans SMTP.

16) L'entête CSeq (ligne 6 abréviation de Command Sequence) définit le numéro de séquence d'une requête. Par exemple ici, on en est très naturellement à la première requête INVITE. L'entête Cseq apparaît aussi dans la réponse. Pourquoi avoir rajouté cette entête ?

17) Définissez un automate de protocole pour la partie SIP client (UAC) concernant uniquement l'ouverture de session INVITE en ne mentionnant sur l'automate que le nom des méthodes (sans les entêtes ni les corps). Vous introduirez les aspects suivants : le client tente sept fois d'ouvrir la session avant d'abandonner. Il peut exister des réponses 100 xxx ou 200 xxx ou des erreurs ou des rejets (400 xxx, 500 xxx). Le message final est un ACK. Indiquez la liste des états que vous utilisez en donnant leur signification.

## Exercice 23 : Protocole UPnP ('Universal Plug and Play')

**Introduction :** UPnP ('Universal Plug and Play') est la réunion de plusieurs protocoles s'exécutant au niveau application de l'architecture de réseaux Internet. Développé depuis 1999, son objectif est de définir des mécanismes, qui permettent à des appareils numériques UPnP, de s'interconnecter automatiquement, c'est-à-dire sans intervention humaine d'administration (d'où le nom 'plug and play'). Un appareil UPnP peut donc, sans intervention humaine, découvrir les autres appareils UPnP accessibles dans son environnement réseau, découvrir les services qu'ils peuvent rendre et se connecter à ces appareils pour créer un réseau à la demande. On parle alors de réseau « ad'hoc » c'est-à-dire d'un réseau de circonstance créé par la mise en présence, d'un ensemble d'appareils qui peuvent communiquer.

**Domaine d'application :** Le domaine des appareils UPnP est pour l'instant celui de l'informatique individuelle et domestique: ordinateurs personnels, téléphones portables, organiseurs, appareils audio/vidéo (décodeurs de télévision, lecteurs de disques, amplificateurs, projecteurs), appareils photos mais aussi appareils ménagers, appareils de domotique (alarmes, chauffage)... Pour exécuter UPnP, ces dispositifs doivent donc disposer d'un minimum 'd'intelligence': un processeur avec des fonctions systèmes de base, un coupleur de communication réseau et une pile de protocoles Internet. Un exemple type d'environnement système en Java pour un appareil est donné par OSGi ('Open Service Gateway initiative') qui est un complément naturel de UPnP. Un exemple type de réseau local utilisable en relation avec UPnP est celui du réseau sans fil Wifi. Cependant, tout type de réseau filaire ou sans fil avec la pile des protocoles Internet peut permettre d'utiliser UPnP: Ethernet, réseau téléphonique avec ou sans fil, infrarouge, à courants porteurs, IEEE1394 Firewire, réseau radio fréquences Bluetooth etc.... Dans la suite de ce texte, nous considérons que c'est un réseau local sans fil Wifi qui supporte les communications aux niveaux physique et liaison.

Une utilisation typique de ce protocole est alors de fournir à un opérateur humain la liste des différents appareils qu'il peut atteindre dans son environnement réseau et lui permettre d'activer différents services offerts par ces appareils au moyen d'une interface ergonomique. Par exemple un téléphone portable (ou un ordinateur ou une télécommande universelle UPnP) peuvent être utilisés comme point de contrôle UPnP permettant de télécommander des appareils périphériques. Un utilisateur à son domicile, pourrait commander la fermeture des volets électriques d'une pièce où il se trouve, régler la température de la pièce, lancer un lecteur de flux vidéo vers le réseau et de manière coordonnée lancer la projection du flux vidéo à partir du réseau. Il pourrait enfin commander l'extinction de la lumière.

**Description générale :** UPnP propose des mécanismes pour :

- l'autoconfiguration des appareils (sans intervention humaine), en particulier tout ce qui concerne la configuration automatique des adresses.
- la possibilité pour un appareil UPnP d'annoncer aux autres sa présence dans un réseau. Cette annonce est diffusée afin d'atteindre tous les destinataires UPnP potentiels.
- la découverte automatique d'autres appareils UPnP, s'ils sont connectés au même réseau. Une requête de découverte, (une demande vers tous les autres appareils UPnP de signaler leur existence) est également diffusée.
- la transmission vers les autres appareils UPnP des différents services que peut rendre un appareil (en fait la description des différentes interfaces applicatives API que peut posséder cet appareil). De même, un appareil doit pouvoir obtenir sur demande les différents services des autres appareils. Il doit pouvoir interpréter et utiliser ces définitions de services. Les possibilités d'annonce de la présence d'un appareil et des services qu'il peut rendre, ou d'interrogation des autres appareils

sur leur présence et leurs services, constituent la principale contribution de UPnP associée à la découverte de la configuration d'un réseau.

- la mise en connexion entre appareils, le contrôle commande à distance d'un appareil. UPnP utilise pour le contrôle commande d'un appareil, les services Webs avec les protocoles SOAP ('Simple Object Access Protocol' et le langage de description d'interfaces WSDL 'Web Service Description Language').

- la possibilité de générer et de recevoir des événements (des messages exceptionnels asynchrones) signalant des changements de variable d'état (changement de mode de fonctionnement ou problèmes sur un appareil). UPnP utilise un protocole de gestion d'événements baptisé GENA ('General Event Notification Architecture').

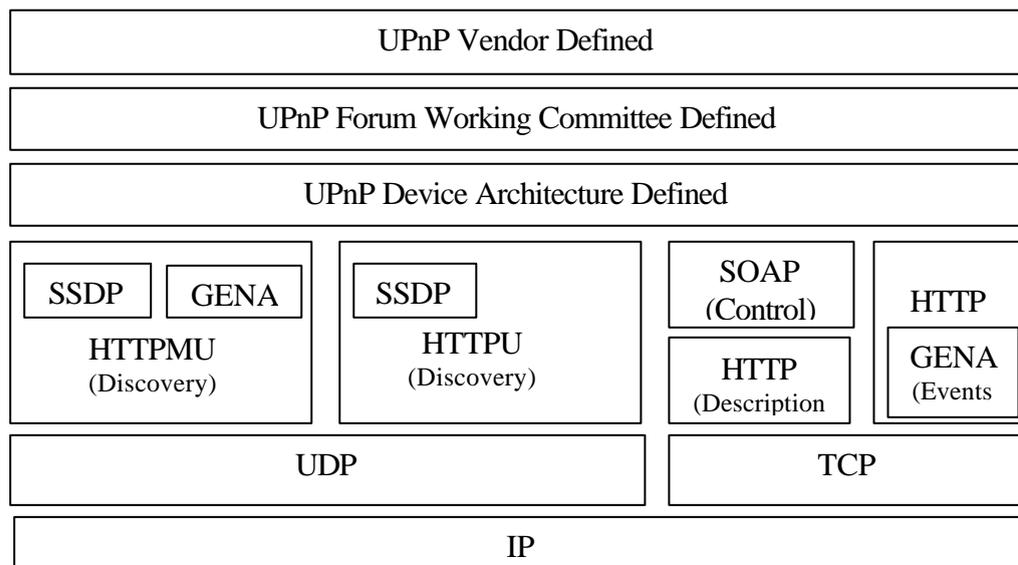
UPnP s'appuie sur des standards Internet très répandus avec principalement HTTP et XML. Pour l'activation des services distants d'un appareil, UPnP propose l'utilisation des services Webs avec SOAP et WSDL. Les protocoles UDP et TCP fournissent l'infrastructure de communication de niveau transport (voir le diagramme d'ensemble des protocoles UPnP figure 1).

Les communications en UPnP sont structurées en mode client-serveur.

- Un appareil client est appelé en UPnP un point de contrôle (en anglais CP 'Control Point). Très souvent, il contrôle pour un utilisateur final (une personne) différents périphériques au moyen d'une interface graphique. Le logiciel UPnP point de contrôle est donc typiquement exécuté sur un ordinateur personnel, un téléphone portable, un organiseur ou une télécommande.

- Un appareil serveur est appelé en UPnP un périphérique (en anglais CD 'Controlled Device'). Un périphérique offre un ou plusieurs services susceptibles d'être découverts et activés par un point de contrôle.

**Diffusion industrielle :** UPnP est soutenu par un consortium de plus de 800 entreprises ou organisations (UPnP Forum) avec un rôle moteur joué par Microsoft qui a installé UPnP en standard dans Windows XP. Dans la version Vista, UPnP a été rebaptisé 'Network Discovery'. Il existe plusieurs autres propositions concurrentes de protocoles différents qui résolvent plus ou moins le même problème : Jini dans l'univers SUN/Java, Salutation, SLP ('Service Location Protocol'), Zeroconf....



**Figure 1.** Pile des protocoles UPnP

## Problèmes d'adressage

Les communications en UPnP sont dérivées des communications en http de sorte que les structures de deux protocoles sont très voisines. On étudie les aspects d'adressage en UPnP en commençant par revenir sur l'adressage en HTTP. Pour communiquer en HTTP, un client HTTP doit connaître ou obtenir par un moyen ou un autre différents noms ou adresses le concernant et concernant le serveur auquel il s'adresse.

1) Rappelez les éléments principaux du système d'adressage utilisé par un client HTTP pour désigner un serveur HTTP. Indication : vous donnerez son nom. Puis vous définirez les différentes informations de désignation (noms et adresses) obligatoires ou facultatives que le client HTTP (l'hôte émetteur d'une requête HTTP) doit connaître pour ce qui concerne le serveur. Ces adresses ou ces noms sont associés aux différentes couches de l'architecture OSI qui sont également celles de l'Internet (aux différents protocoles Internet). Vous en donnerez une liste complète classée par couches, en précisant les noms ou adresses qui sont obligatoires et ceux qui sont optionnels c'est-à-dire obtenus par des valeurs par défaut ou par des moyens divers automatiques (valeurs pré configurées ou obtenues par utilisation de protocoles).

2) Quelles sont les différentes informations de désignation (adresses ou noms symboliques) que le client HTTP (l'hôte émetteur d'une requête HTTP) doit connaître pour ce qui le concerne. Mêmes indications que pour la question précédente. Les adresses ou les noms sont à analyser niveau par niveau. Ces adresses ou ces noms concernent le client en distinguant les noms ou adresses obligatoirement fournies de ceux ou celles qui sont optionnelles.

3) On se place maintenant dans le cas du protocole UPnP, dont le but est d'auto configurer un réseau composé d'appareils d'informatique individuelle ou domestique, en utilisant des communications de type HTTP. Les adresses ou les noms utilisés sont de même nature que ceux de HTTP. Dans certains cas le réseau peut-être correctement administré avec des serveurs caractéristiques d'un bon réseau d'entreprise (serveurs DNS, DHCP). Dans d'autres cas le réseau dans lequel se trouvent les appareils UPnP n'est pas du tout administré et il n'existe aucun serveur pré installé. De plus les appareils UPnP peuvent être par nature très simples (comme une commande d'éclairage d'une pièce). Dans le cas d'un protocole comme UPnP, quels sont les noms ou adresses qui sont indispensables ? Indications : vous utiliserez les réponses aux questions précédentes concernant le client et le serveur en vous préoccupant principalement des noms ou adresses indispensables pour atteindre un serveur HTTP distant. Vous proposerez éventuellement des solutions pour obtenir ces noms ou adresses.

## Découverte des services disponibles dans un réseau : SSDP (‘Simple Service Discovery Protocol’)

SSDP est dans l'ensemble UPnP, le protocole qui permet à un appareil UPnP de détecter tous les autres appareils UPnP connectés à un réseau. Pour découvrir un réseau, un appareil peut émettre des requêtes lui permettant de connaître la liste des appareils déjà connectés et pour chaque appareil la liste des services qu'il peut fournir. Par ailleurs un appareil qui vient d'être initialisé doit envoyer des requêtes annonçant son arrivée dans le réseau et définissant les services qu'il peut fournir. Quand un appareil est arrêté normalement, il doit émettre une requête indiquant sa sortie du réseau. En ce sens SSDP entre dans la catégorie des protocoles d'appartenance à un groupe dont le rôle est de suivre la composition d'un groupe en univers réparti ('Group Membership Protocol').

Avec SSDP, pour chaque appareil, le suivi de la composition du groupe des appareils accessibles doit être réalisé tant que l'appareil reste opérationnel. Donc, pour maintenir sa vision à jour après la phase initiale d'apprentissage, chaque appareil SSDP émet des requêtes de découverte, traite les réponses et reçoit périodiquement des messages d'annonce SSDP ce qui génère un flux de messages permanent.

SSDP est une adaptation de HTTP au problème de la découverte de réseau. Il conserve de HTTP sa structure générale, ses formats de messages, mais, comme ses objectifs sont différents il utilise des méthodes nouvelles et des mécanismes nouveaux.

4) Quels sont les avantages et les inconvénients de prendre comme base le protocole HTTP pour un nouveau protocole (par exemple ici un protocole de découverte de réseau comme SSDP) ?

Une caractéristique importante de SSDP est d'utiliser deux versions modifiées de HTTP qui ont été baptisées HTTPU pour 'HTTP Unicast over UDP' et HTTPMU pour 'HTTP Multicast over UDP'.

Une différence importante entre HTTPU et HTTP est donc que HTTPU utilise le transport UDP comme protocole de transport sous jacent et non pas TCP. Cependant, comme en HTTP, les communications en HTTPU sont point à point entre un seul client et un seul serveur.

Le protocole HTTPMU étend HTTP à des communications en diffusion sur groupe ('multicast'). C'est-à-dire que pour un client, il existe n destinataires d'une requête. Ces destinataires appartiennent à un groupe (un sous ensemble des sites, par exemple ici ceux qui sont UPnP). Selon les cas, tous les destinataires ou un sous ensemble des destinataires ou aucun destinataire fournissent une réponse à la requête. En HTTPU comme en HTTPMU les requêtes et les réponses ont un format similaire à celui de HTTP. HTTPMU utilise également comme protocole de transport sous-jacent UDP.

5) Pourquoi HTTP utilise t'il TCP comme protocole de transport (quels sont les avantages et les inconvénients du choix de TCP comme protocole de transport sous jacent au protocole HTTP)?

6) Quelles sont les raisons qui motivent le concepteur d'un protocole de découverte de réseau comme SSDP à substituer UDP comme transport sous jacent à TCP. Indication, examinez les caractéristiques que vous connaissez des deux protocoles TCP et UDP en relation avec les objectifs poursuivis par SSDP ?

Le protocole HTTPU a reçu un nom de schéma d'adressage dans le cadre des URL, différent de HTTP. C'est naturellement HTTPU qui a été retenu. Une URL HTTPU commence donc par la chaîne de caractères 'HTTPU:'. En dehors du changement de nom de schéma, les URL HTTPU ont sensiblement la même structure (la même grammaire) que les URL HTTP.

7) Puisque la structure des URL est la même, pourquoi ne pas avoir conservé comme nom de schéma HTTP (quelles sont les raisons qui motivent le changement du nom de schéma) ?

Dans le protocole SSDP, **NOTIFY** est le nom de l'une des méthodes (comme GET, HEAD ou POST en HTTP). Une requête NOTIFY est toujours transmise en diffusion au moyen de HTTPMU à tous les autres appareils UPnP accessibles. NOTIFY est décrit dans la norme UPnP comme un avertissement ('advertisement') en ce sens que cette méthode avertit les autres appareils UPnP de l'existence de l'émetteur de la requête NOTIFY. Une telle requête est donc naturellement transmise à la mise sous tension d'un appareil ou lors d'une modification de configuration ou encore périodiquement pour rafraîchir les caches des autres appareils. Dans la terminologie UPnP, le cache

est en fait la table gérée par chaque appareil qui représente sa vision de l'ensemble des autres appareils. L'émetteur d'un NOTIFY apporte simplement une information aux autres et n'attend aucune réponse des autres appareils.

La description des principales entêtes d'une requête NOTIFY est donnée dans la liste suivante (extraite de la norme UPnP) :

**NOTIFY** \* HTTP/1.1  
**HOST:** 239.255.255.250:1900  
**CACHE-CONTROL:** max-age = seconds until advertisement expires  
**LOCATION:** URL for UPnP description for root device  
**NT:** notification type  
**SERVER:** OS/version UPnP/1.0 product/version  
**NTS:** ssdp:alive ou ssdp:byebye  
**USN:** unique service name

Explications complémentaires:

**LOCATION :** Définit une URL du WEB où l'on peut trouver une ressource (une page) contenant une description détaillée de l'appareil émetteur de la requête. C'est cette information qui est essentielle car cette description peut être téléchargée de manière à obtenir les informations nécessaires au contrôle à distance d'un appareil.

**NT ('Notification Type') :** Définit le type de l'information apportée. Un appareil ('device') peut comporter des sous appareils (sub devices), chacun pouvant offrir des services différents. Le type dans un avertissement permet de séparer ce qui concerne l'appareil principal (root device) ou l'un des sous appareils ou l'un des services de cet appareil ou l'un des services de l'un des sous appareils.

**SERVER :** identifie le nom du système d'exploitation (Linux, Windows...) de l'appareil et sa version. On trouve ensuite la version de UPnP utilisée (actuellement 1.0) puis le nom et la version donné par le constructeur pour l'appareil qui effectue l'annonce.

**NTS 'Notification Sub Type' :** Dans cette entête, la chaîne ssdp:alive indique que l'appareil ou le service est opérationnel, ssdp:byebye indique l'arrêt du fonctionnement opérationnel d'un appareil ou de l'un de ses services quand il en offre plusieurs.

**USN 'Unique Service Name' :** c'est un identifiant unique pour un service.

Voici un exemple concret de requête NOTIFY, extrait d'une documentation Microsoft :

```
NOTIFY * HTTP/1.1\r\n
Host:239.255.255.250:1900\r\n
NT:urn:schemas-upnp-org:service:ConnectionManager:1\r\n
NTS:ssdp:alive\r\n
Location:http://192.168.1.100:2869/upnphost/udhisapi.dll?content=
        uuid:224e2bb9-6961-4d79-b05f-f72cb415dc6c\r\ns
USN:uuid:224e2bb9-6961-4d79-b05f-f72cb415dc6c::urn:schemas-upnp-
org:service:ConnectionManager:1\r\n
Cache-Control:max-age=1800\r\n
Server:Microsoft-Windows-NT/5.1 UPnP/1.0 UPnP-Device-Host/1.0\r\n
\r\n
```

8) Dans la requête NOTIFY, on voit dans les deux premières lignes les aspects d'adressage dans une forme qui est celle de HTTP 1.1. Associé au nom de la méthode on a le chemin vers la ressource et la seconde ligne est une entête HOST. Pourquoi en HTTP/1.1 a-t-on introduit de manière obligatoire l'entête HOST ?

9) Le chemin vers la ressource est défini par \*. L'entête HOST a pour valeur 239.255.255.250:1900. Pourquoi en SSDP a-t-on défini l'adresse cible de la requête de cette

façon ? Indication : l'objectif est celui d'une diffusion vers tous les appareils UPnP d'un avertissement et on a souhaité respecter les conventions de HTTP/1.1 ?

10) Il existe une ligne d'entête CACHE-CONTROL dont le paramètre principal max-age définit la durée en seconde jusqu'à expiration de l'avertissement (c'est à dire de l'information apportée par la requête NOTIFY). Pourquoi définir une telle durée de validité dans les caches ?

M-SEARCH est une seconde méthode définie en SSDP pour enquêter sur la présence d'appareils d'un certain type. Elle est comme NOTIFY transmise en diffusion HTTPMU à tous les autres appareils UPnP.

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: seconds to delay response
ST: search target
```

L'entête MAN contient la chaîne ssdp:discover pour indiquer une requête de découverte de configuration réseau.

Une réponse positive, obtenue lorsqu'une information est retournée correctement sur une requête M-SEARCH, a la forme suivante. C'est une réponse point à point transmise en HTTPU. Les entêtes ont la même signification que dans notify.

```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age = seconds until advertisement expires
LOCATION: URL for UPnP description for root device
ST: search target
USN: unique service name
```

11) On remarque parmi les entêtes une entête nouvelle MX obligatoire définissant une valeur en secondes. Un appareil qui répondra à une requête M-SEARCH doit attendre un délai distribué aléatoirement entre 0 et MX secondes pour envoyer sa réponse. Quand un réseau UPnP augmente en taille, la norme recommande d'augmenter progressivement la valeur de MX. Selon vous, dans quel but l'entête MX a-t-elle été définie ? Eventuellement définissez à quelles classes de mécanisme dans les réseaux, correspond l'entête MX.

## Spécification d'un appareil et de ses services

Il existe différents documents XML pour la description des appareils et de leurs services : description physique d'ensemble d'un appareil, description logique, description détaillée des services rendus, description des variables d'état qu'un appareil gère, description des événements générés. Le principal document (la description d'ensemble) a la forme générale suivante :

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
<URLBase>base URL for all relative URLs</URLBase>
<device>
<friendlyName>short user-friendly title</friendlyName>
```

```

<manufacturer>manufacturer name</manufacturer>
<manufacturerURL>URL to manufacturer site</manufacturerURL>
<modelDescription>long user-friendly title</modelDescription>
<modelName>model name</modelName>
<modelNumber>model number</modelNumber>
<modelURL>URL to model site</modelURL>
<serialNumber>manufacturer's serial number</serialNumber>
<UDN>uuid:UUID</UDN>
<UPC>Universal Product Code</UPC>
<deviceType>urn:chemas-upnp-or:device:deviceType</deviceType>
<serviceList>
<service>
<serviceType>urn:schemas-upnp-org:service:serviceType:v</serviceType>
<serviceId>urn:upnp-org:serviceId:serviceID</serviceId>
<SCPDURL>URL to service description</SCPDURL>
<controlURL>URL for control</controlURL>
<eventSubURL>URL for eventing</eventSubURL>
</service>
    Declarations for other services (if any) go here
</serviceList>
<deviceList>Description of embedded devices (if any) go here</deviceList>
<iconList>
<icon>
<mimetype>image/format</mimetype>
<width>horizontal pixels</width>
<height>vertical pixels</height>
<depth>color depth</depth>
<url>URL to icon</url>
</icon>
    XML to declare other icons, if any, go here
</iconList>
<presentationURL>URL for presentation</presentationURL>
</device>
<specVersion><major>1</major><minor>0</minor></specVersion>
</root>

```

12) Lecture du document XML. En vous limitant aux quatre premières lignes du document, donnez des explications concernant ces lignes. Indications : commentez les entêtes, les éléments, les attributs. Quelles sont les informations apportées par ces quatre lignes ?

13) Quel est l'arbre associé à l'élément 'servicelist'. Que reflète cette structure (pourquoi avoir retenu cette structure) ?

14) Dans une interface graphique ergonomique, à chaque appareil on associe une icône dont le graphisme doit être familier à l'utilisateur. Quels sont les éléments qui ont été retenus pour préciser l'une des icônes associée à un appareil (pouvez vous expliquer leur signification).

## UPnP et services d'annuaires

15) Une autre approche possible pour résoudre le problème de découverte de la configuration d'un réseau résolu par UPnP est d'utiliser une approche d'annuaire comme celle du DNS ('Domain Name System'). Mais de manière similaire les annuaires LDAP ('Lightweight Directory Access Protocol') ou UDDI des services Webs ('Universal Description Discovery and Integration') seraient aussi utilisables. Rappelez brièvement les principes généraux d'architecture et les principales fonctions d'un service d'annuaire ?

16) Comment pourrait-on utiliser un service d'annuaire pour fournir un service de découverte de la configuration d'un réseau ? On rappelle qu'un appareil peut annoncer sa présence et envoyer des requêtes de découverte des autres appareils accessibles et des services qu'ils peuvent rendre.

17) Discutez des avantages et des inconvénients d'une approche complètement répartie ('pair à pair' 'peer to peer') comme celle de UPnP ou l'on fait jouer le même rôle à tous les appareils plutôt que de faire gérer ces informations par un service d'annuaire centralisé. En UPnP chaque appareil gère ses propres caractéristiques, les fournit sur demande. Il apprend les caractéristiques de tous les autres appareils. Indications : vous pourrez examiner les aspects d'administration, de performances, de tolérances aux pannes, de conception et de développement de l'application.

### *Complément hors sujet*

**Vocabulaire :** UPnP participe à la construction d'un réseau **ad'hoc** c'est à dire un réseau qui s'auto organise selon les machines en présence. L'approche UPnP est développée entre des appareils considérés comme égaux. UPnP entre donc dans la catégorie des approches '**pair à pair**' ('**peer to peer**'). Par ailleurs, UPnP est l'un des moyens de l'informatique **omniprésente** ou **ambiante** ou encore **ubiquitaire**, c'est-à-dire présente dans de plus en plus d'aspects (des appareils) de la vie courante. On parle en anglais d'informatique '**pervasive**' (en français ominiprésente) ou encore '**ubiquitous**' (ubiquitaire) ou encore '**ambient**' (ambiante). Les personnes évoluent de plus en plus dans une ambiance de technologie informatique et réseau.