Sujets des exercices dirigés

Réseaux et communication

Seconde partie

2002-2003

G. Florin, E. Gressier, S. Natkin

Table des matières

PREMIER CHAPITRE

EXERCICES CONTROLE REPARTI

Série d'exercices mode message

Exercice 1: Automate de connexion TCP

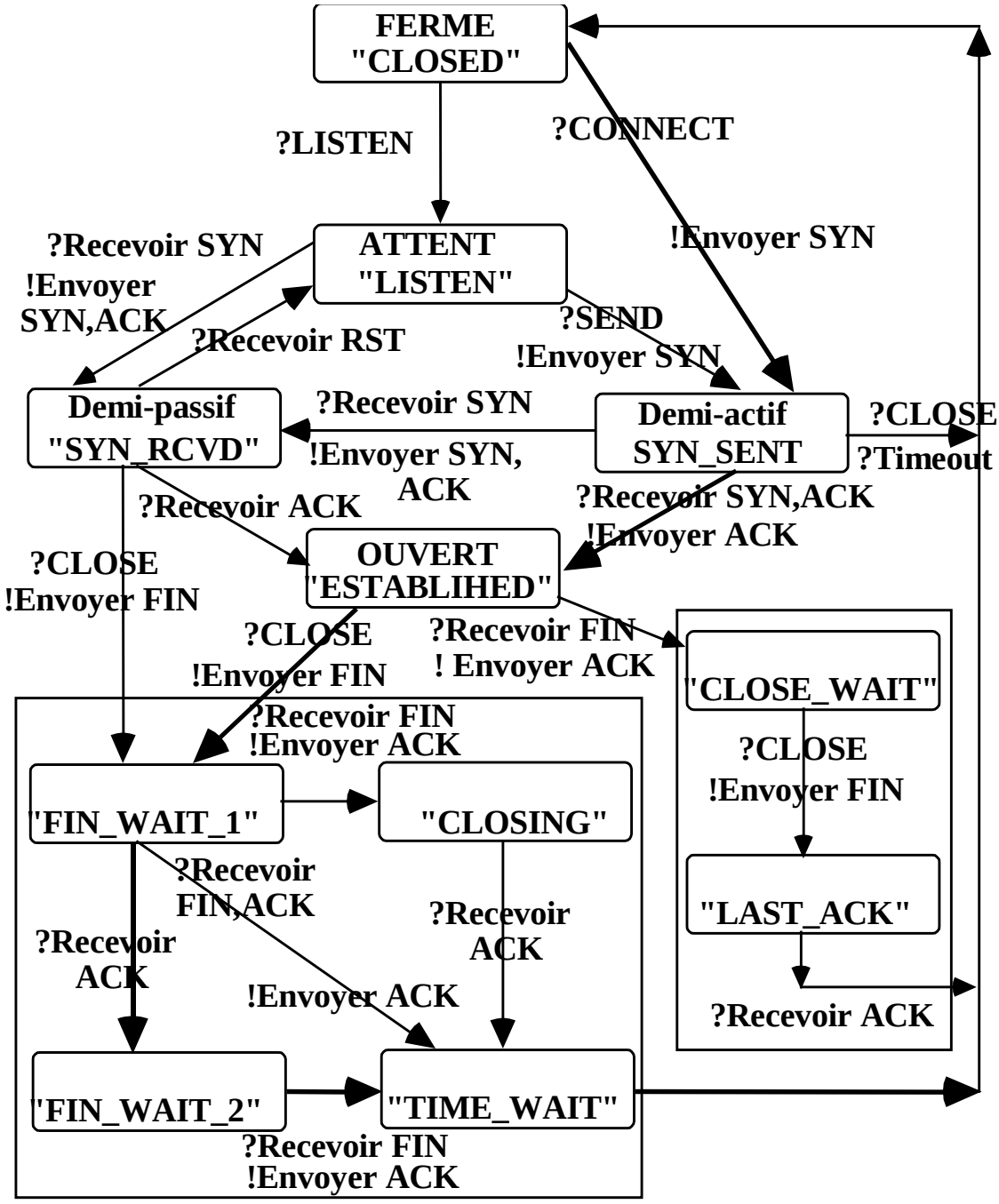
1) Citez différentes raisons qui justifient l'introduction de la couche transport dans la pile des protocoles de communication OSI.

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 donné 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, …).


FERME
"CLOSED"
?LISTEN
?CONNECT
?Recevoir SYN
!Envoyer
SYN,ACK
ATTENT
"LISTEN"
!Envoyer SYN
?SEND
!Envoyer SYN
?Recevoir RST
Demi-passif
"SYN_RCVD"
?Recevoir SYN
!Envoyer SYN,
ACK
Demi-actif
SYN_SENT
?CLOSE
?Timeout
?Recevoir ACK
?Recevoir SYN,ACK
!Envoyer ACK
?CLOSE
!Envoyer FIN
OUVERT
"ESTABLIHED"
?CLOSE
!Envoyer FIN
?Recevoir FIN
! Envoyer ACK
"CLOSE_WAIT"
?Recevoir FIN
!Envoyer ACK
?CLOSE
!Envoyer FIN
"FIN_WAIT_1"
"CLOSING"
?Recevoir
FIN,ACK
?Recevoir
ACK
?Recevoir
ACK
!Envoyer ACK
"LAST_ACK"
?Recevoir ACK
"FIN_WAIT_2"
"TIME_WAIT"
?Recevoir FIN
!Envoyer ACK

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 considére qu'il n'y a pas de partage de données entre plusieurs activités?

Exercice 5 : Serveurs multiples (redondance sélective)

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

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

Proposez une organisation pour rendre efficacement un tel service.

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 entr'elles) 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 calculateur émetteur E sur un calculateur 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 coté 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 : Programmation d'applications réparties en mode message et en appel de procédure distante.

Le problème consiste à étudier l'expression d'un protocole assez simple en mode message asynchrone puis en mode appel de procédure distante. L'application considérée est l'ouverture de connexion à trois étapes.

On rappelle le protocole d'établissement d'une connexion en point à point à trois étapes ("three way handshake"). Cette solution est utilisée dans les protocoles de transport comme TP4 ou TCP et utilise donc le mode de communication par message asynchrone. Pour simplifier la question on ne traite que le cas unidirectionel ou un site demandeur propose à un autre site accepteur l'établissement d'une connexion. Egalement on ne considère ni les pertes de messages ni les réponses tardives.

L'application que nous considérons comporte donc deux entités dénommées le demandeur (de connexion) et l'accepteur (de connexion). Sur chacun des sites (du demandeur et de l'accepteur) existent des entités prestataires de service qui réalisent le service d'établissement de connexion. Entre les deux prestataires de service est réalisé le protocole proprement dit de connexion.

Le dialogue entre l'usager demandeur et le prestataire du service demandeur est réalisé par trois unités de données de service:

- La demande initiale avec paramètres de qualité de service du demandeur
connexion_requète (<paramètres_D>)
- La réponse de confirmation d'ouverture avec la contre_proposition de l'accepteur concernant la qualité de service
connexion_confirmation (< paramètres_A>)
- La réponse de rejet d'ouverture
deconnexion_indication (<>)

Le dialogue de l'usager accepteur et de son prestataire local est réalisé par trois unités de service (SDU):

- L'arrivée d'une demande de connexion
connexion_indication (<paramètres_D>)
- La réponse positive à la proposition d'ouverture avec contre proposition de qualité de service. Celle-ci est une réduction de la proposition du demandeur et est une proposition définitive.
connexion_reponse (< paramètres_A>)
- La réponse négative : rejet de l'ouverture
deconnexion_requète (<>).

On appelle X la référence unique (le nom) de la connexion en cours d'ouverture pour le site demandeur et Y la référence unique pour le site accepteur On peut imaginer que les références de connexion pour chaque site sont les numéros des entrées dans les tables des connexions ouvertes sur chaque site. L'ensemble des deux couples (émetteur,X) (récepteur,Y) identifie complètement la connexion. Les unités de protocole (PDU "protocol data unit" ou messages) qui peuvent être échangés sont les suivants:

- Le message qui véhicule la demande de connexion

CR(X, <paramètres_D>)

A l'arrivée de ce message l'accepteur sait que le demandeur souhaite ouvrir une connexion sous la référence X.

- Le message qui véhicule la réponse d'acceptation de l'ouverture.

CC(X,Y, <Paramètre_A>)

A l'arrivée de ce message le demandeur sait que l'accepteur accepte l'ouverture et que la référence de connexion est X,Y)

- Le message qui véhicule le rejet de la demande de connexion par l'accepteur (c'est ce message qui est envoyé quand l'accepteur refuse l'ouverture de la connexion)

DR(X)

- Le premier message de données: message indispensable pour terminer l'ouverture de connexion:

DT(X,Y, <>)

A l'arrivée de ce message l'accepteur sait que l'ouverture de connexion s'est bien passée (que le demandeur sait que l'accepteur a accepté la demande).

Question 1 Dans un premier temps on spécifie le comportement en utilisant le mode message. On utilise donc une technique à automate d'états avec des conditions de franchissement des transitions, des émissions de SDU ou de PDU notés !<PDU ou SDU>, des réceptions en rendez-vous de PDU ou de SDU notés ?<PDU ou SDU>)

1.1Construisez l'automate de service de l'usager demandeur (l'ensemble des comportements autorisés de l'usager demandeur (automate ou vous ne portez que les SDU).

1.2 Construisez l'automate de service de l'usager accepteur

1.3 Construisez l'automate complet de comportement du prestataire du service réseau du site du demandeur. Cet automate comporte les interactions avec l'usager local au moyen des unités de service et les interactions avec le prestataire distant au moyen de l'échange de PDU.

1.4 Construisez l'automate complet du comportement du prestataire du service réseau du site accepteur.

Les solutions présentées doivent être analysées le plus précisément possible afin que votre solution soit compréhensible. En particulier chaque état identifié doit recevoir une signification simple. Chaque transition également. Toutes les possibilités doivent être étudiées. Vous pourrez vous aider de diagrammes d'échanges de SDU et de PDU mais ces diagrammes ne correspondent qu'a une seule situation et ne remplacent pas un automate.

Question 2 On suppose maintenant l'existence d'un appel de procédure distante utilisable pour l'implantation de l'ouverture de connexion. L'appel de procédure est réalisé par un objet qui comporte des procédures éxécutables selon des appels locaux et des procédures utilisables en appel distant. Chacun des deux sites communicants comporte une instance de l'objet de connexion.

Décrivez l'organisation générale d'une ouverture de connexion réalisant les mêmes fonctions qu'à la question précédente (on décrira brièvement les fonctions à réaliser par les différentes procédures à réaliser).

Conseil pour la solution: établir une analogie entre les messages utilisés dans la spécification en termes de rendez-vous et les messages véhiculés en appel et retour d'exécution de procédure distante.

Exercice 9 : 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 ou des erreurs auraient rendu l'opération impossible. Les structures de données échangées dans les messages en langage C sont donc décrites comme suit :

```
#define MAXSTOCKNAMELEN 100

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

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

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

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

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

```
typedef int HANDLE;

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

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

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

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

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

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

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

3) Quel est le rôle de la primitive connect ?

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

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

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

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

```
strncpy (req.name, stock_name, packet_len);

/* Calcul longueur totale de la requête et conversion vers l'ordre des octets réseau */
packet_len = packet_len + sizeof req.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 ? (1 point)

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) ?

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 ou 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 calculateur 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 ¿b matérialise un espace

" être¿bou¿bne¿bpas¿bêtre¿b"

On mémorise :

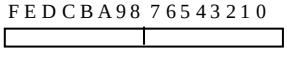
. un mot qui représente l'entier "0014" en hexadécimal pour dire que la chaîne fait 20 caractères

. la chaîne elle-même " être¿bou¿bne¿bpas¿bêtre¿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 

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'abbréviation de Presentation Data Value. L'exercice consiste à commenter cette spécification pour comprendre les choix effectués et par suite préparer correctement des messages.

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

Presentation-context-identifier ::= **INTEGER**
Transfer-syntax-name ::= **OBJECT IDENTIFIER**

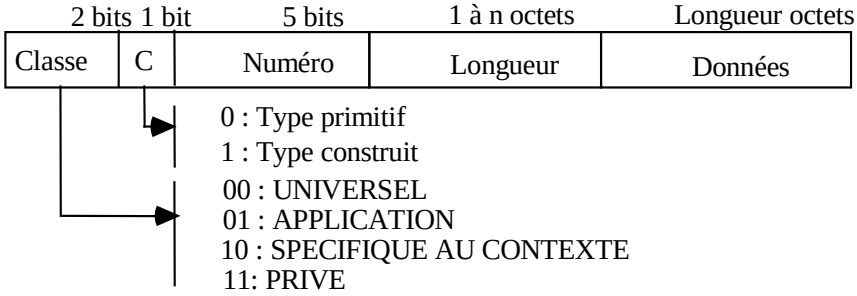
1 Le concepteur a défini deux formats principaux de données utilisateurs. Quels sont-ils? En fonction de leurs définitions a quels besoins correspondent t'ils selon vous?
2 La définition de la syntaxe de transfert utilisée est optionnelle. Pourquoi ce choix est-il effectué?
3 La définition du contexte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

Exercice 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 a jour par des utilisateurs. Nous avons donc un modèle client serveur.

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

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 interfacer 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 tout 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 <> 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 roxrigpv
vco cp xgeyv tb xcojcyb
ib qrcsbox vb xorcsg zyv
ab igyvvb g ebpvbo ig syb
jcb wyobpx qbv tbcd gzyv
ib obfgi wcx wrox mrppbxb
oybp pb zgpjjcgyx gc wbvxyp

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 rpmrn 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 ou 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 mod 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 MlT. 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 $l0^{l0}$?

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

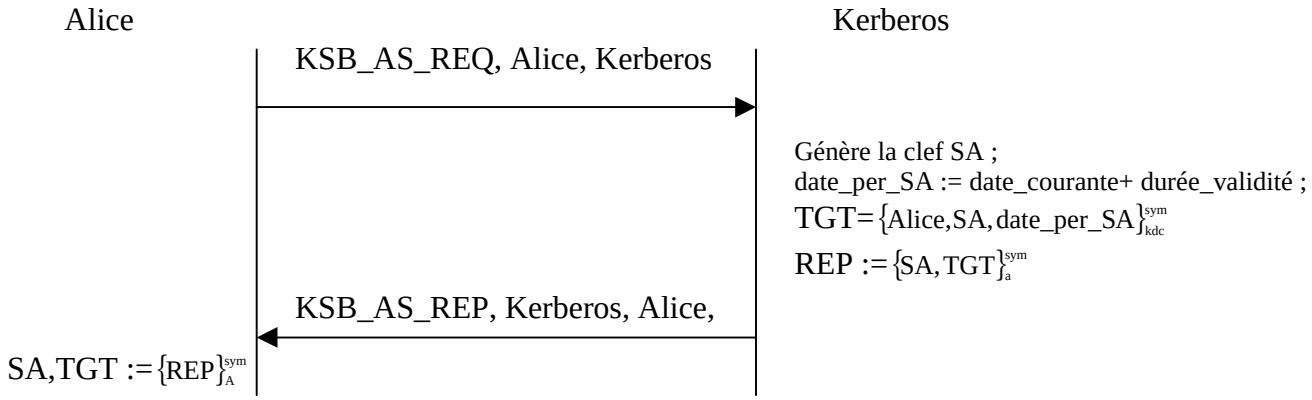
Kerberos met en communication trois entités :

- 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 utilser 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 envoi 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 échange 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.

Exercice15 : 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 b 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 clef 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 a 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 a vis des mêmes intervenants. Expliquez de manière très précise pourquoi votre solution assure les propriétés recherchées.

Question 4 On suppose que l'usager client A dispose d'un délai dmax pour signaler la perte de sa clé. Toute perte non signalée dans le délai dmax ne peut-être tenue à charge contre l'usager. Toute perte non signalée après le délai dmax 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 dmax. 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 calculateur est tombé en panne à ce moment. Comment le notaire peut-il contrer 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. Kb 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----->A:B,A,$\{N\}_{Kb-1}$

A vérifie que $\{\{N\}_{Kb-1}\}_{Kb}$=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 Kb 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 Ks est connue de tous les utilisateurs habilités à utiliser le protocole de sécurité.

Un enregistrement du fichier a la structure suivante:

$B,Kb,\{H(B,Kb)\}_{Ks-1}$

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 Kb 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----->B:A,B,$\{Y,K',Num,H(Y,K',Num)\}_{Kb}$

3) B calcule

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

4) Message n+1: B----->A:B,A,$\{Z\}_{Ka}$

5) A calcule $Z=\{\{Z\}_{Ka}\}_{Ka-1}$

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.

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

1.4.3 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 asychrone 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.

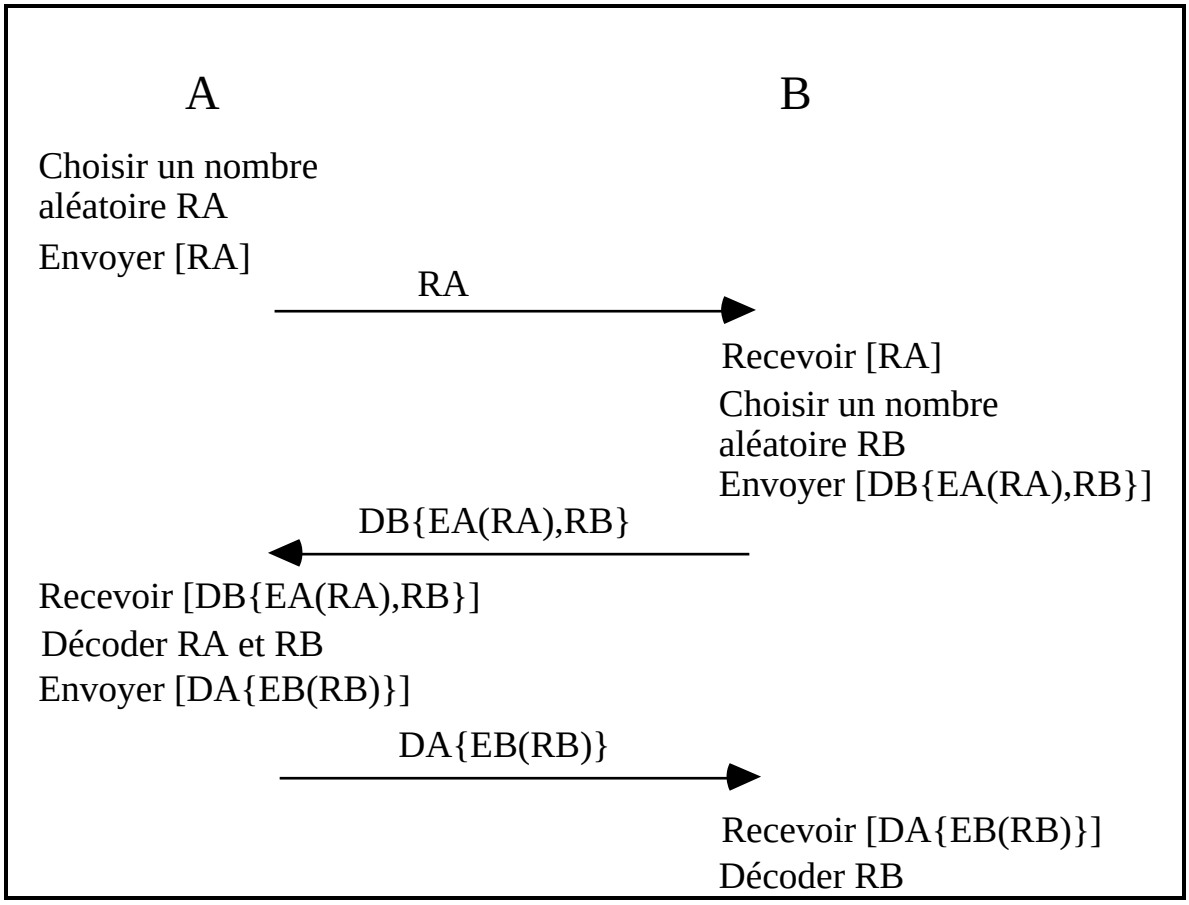
Dans les exemples traités A est l'appelant et B l'appelé.

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 Px pour définir le protocole Py 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 t-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 ?

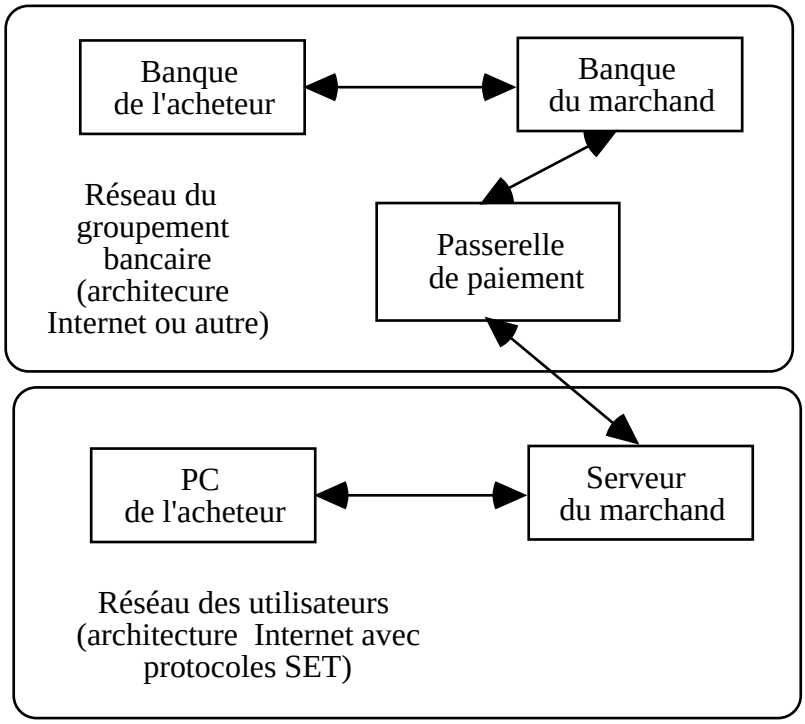
4.2 Quel est l'inconvénient majeur de cette approche ?

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 ou 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 succinte 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

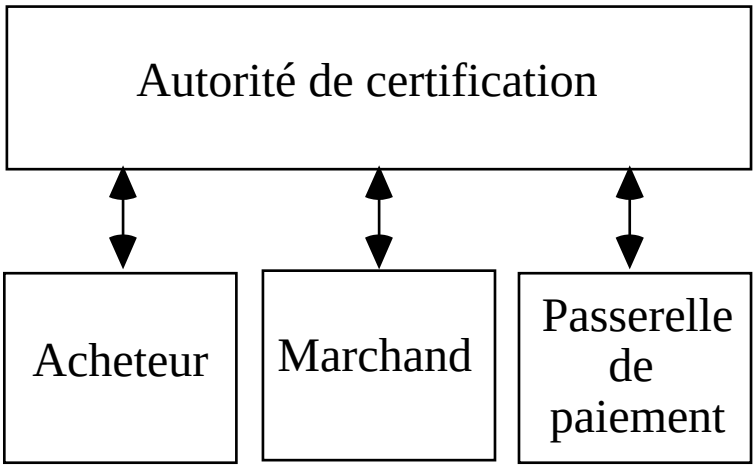
a) La carte bleue
b) Le PIN code de la carte bleue ("Personal Identification Number", le code secret)
c) Le numéro de la carte bleue
d) L'identifiant de la commande
e) Le contenu qualitatif de la commande (sa nature)
f) Le prix de la commande

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.

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 a 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 a 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)

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'usager 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'usager 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 QWxhZGRpbjpvcGVuIHNlc2FtZQ==

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'usager de savoir quel nom d'usager 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'usager daniel.

Il faut bien sur 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 protèger 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'usager daniel pour les opérations GET, PUT et POST , et l'usager laurent uniquement pour les requêtes GET sur ces nouvelles pages. Quels sont les fichiers qui doivent être créés et à quel endroit doivent t'il 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'usager. 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 usager 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 coté navigateur client pour simplifier la vie de l'usager 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'usager et le mot de passe, le navigateur client doit alors répondre par un nom d'usager et à la place du mot de passe la valeur :

MD5(concat (nom d'usager, 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'usager et un mot de passe doit fabriquer un texte qui concatène le nom d'usager, 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'usager, MD5(concat (nom d'usager, 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.

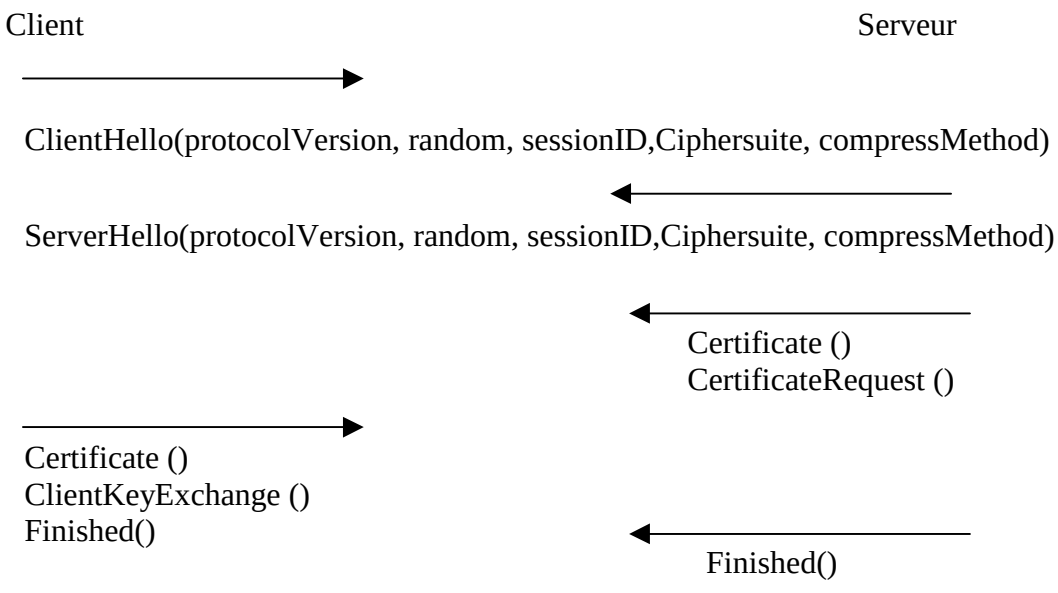
4) Le protocole SSL/TLS propose, dans l'une de ses modalités, d'utiliser la notion de certificat. Cette approche est d'ailleurs de loin la plus souvent retenue. A quoi sert un certificat. Rappelez les principaux champs d'un certificat ?

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.



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 créée 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 coté 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?

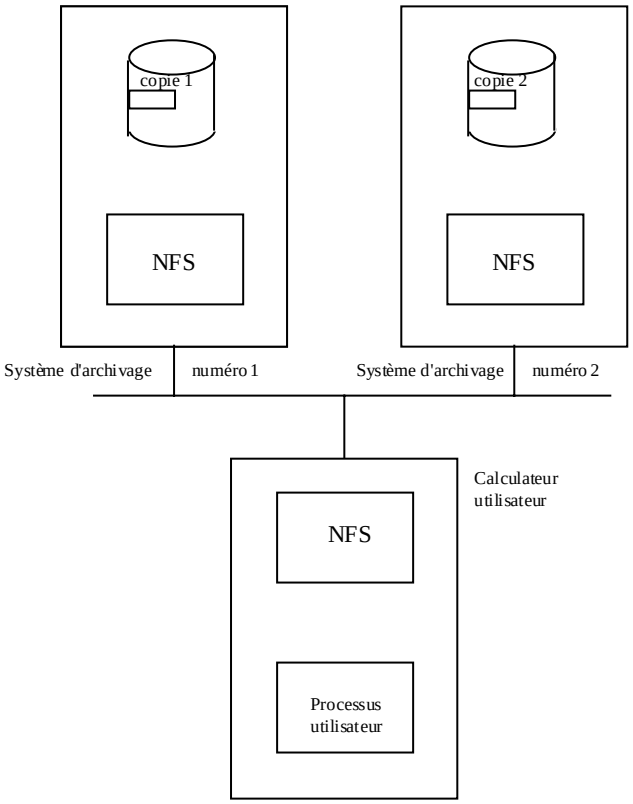
8) Le protocole SSL utilise une combinaison de méthodes de cryptographie qui comprend des algorithmes à clés publiques et des algorithmes à clés privées. Pourquoi utiliser concurremment les deux techniques?

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

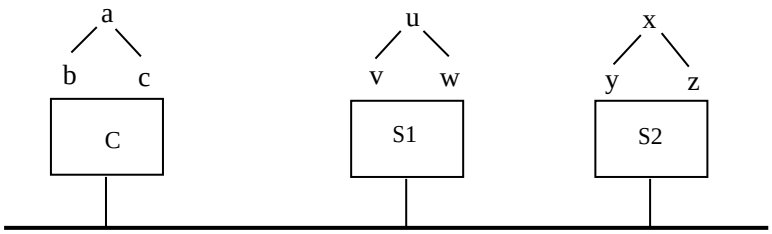
NFS (Network File system) est un système de gestion de fichiers répartis. L'arborescence de fichiers, répartie physiquement sur plusieurs machine et accessible par l'usager depuis son poste de travail, est construite en utilisant une technique de montage d'arborescence. La vision qu'a un utilisateur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

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

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

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

Soit la configuration :



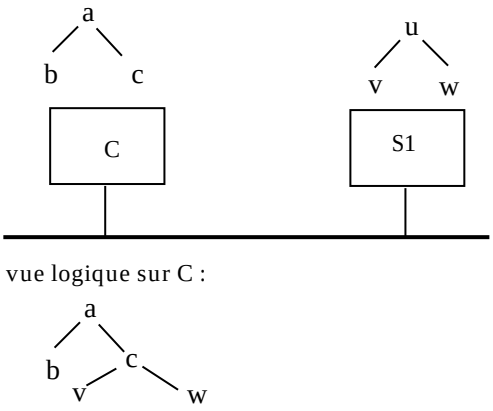
On dispose des opérations suivantes :

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

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

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

donne :



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

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

contenu <chemin d'accès à un répertoire>

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

4 - On considère la suite de commandes 1

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

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

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

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

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

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

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

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

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

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

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

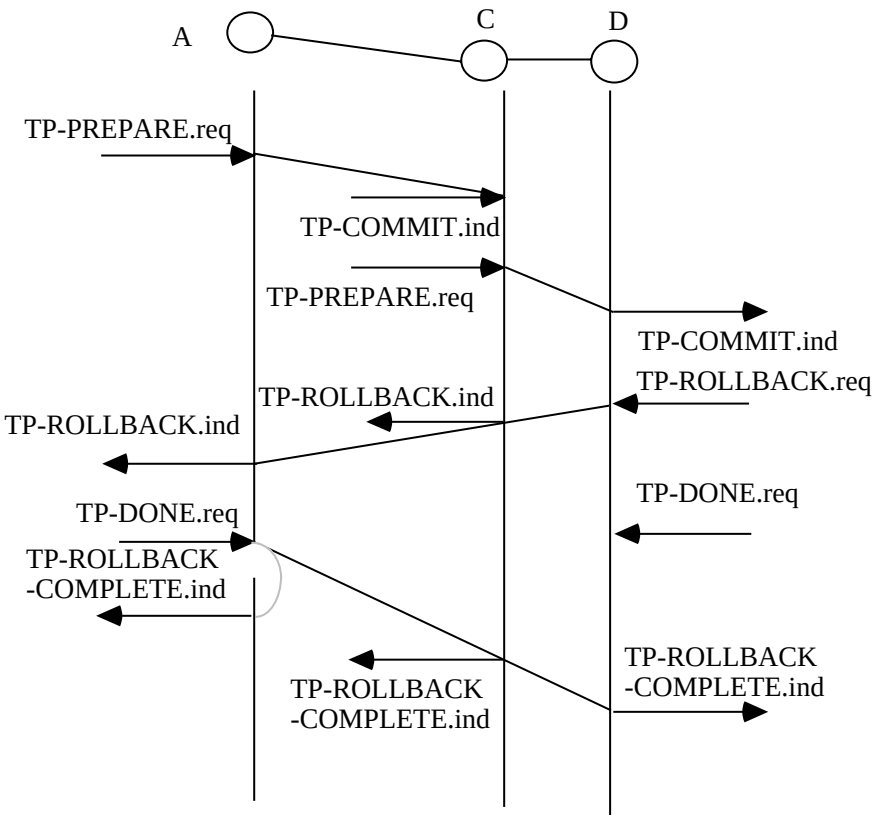
1 Rappelez les principaux problèmes à résoudre pour réaliser des transactions en univers réparti.

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 a flèches comportant la description des éléments de services générés par les trois sites, leur séquencement 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 Ecrit-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 euille les primitives de validation à deux phases permettant de mettre en oeuvre 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?

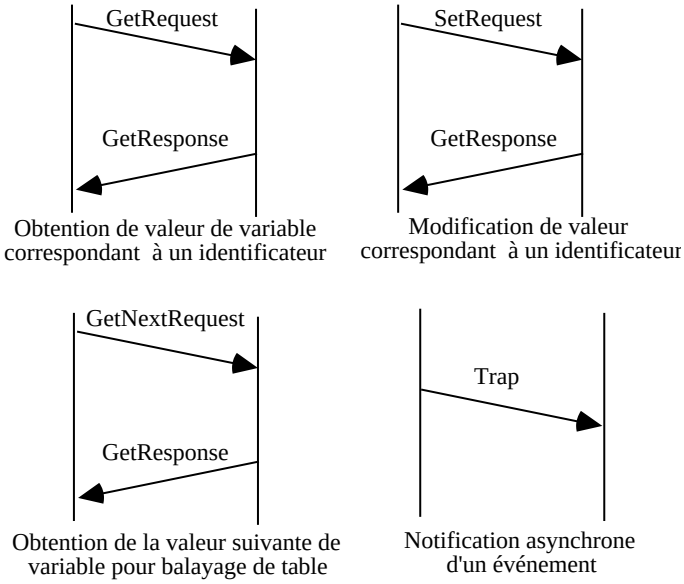
8 - On souhaite représenter sur un automate le mécanisme de traitement des pannes des serveurs. Ajoutez sur l'automate de la question 6 les éléments qui correspondent au traitement de la panne franche des serveurs: délais de garde, traitements à échéance ou l'on suppose que le serveur est en panne (représentez 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 décrire des applications (baptisée station d'administration SNMP) qui vont d'acquérir en mode client/serveur des valeurs caractéristiques du fonctionnement d'un appareil distant. Cet entité réseau est baptisé agent SNMP. Les paramètres lus ou positionnés sur l'agent SNMP sont appelés variables ou objets dans la suite.

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

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



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

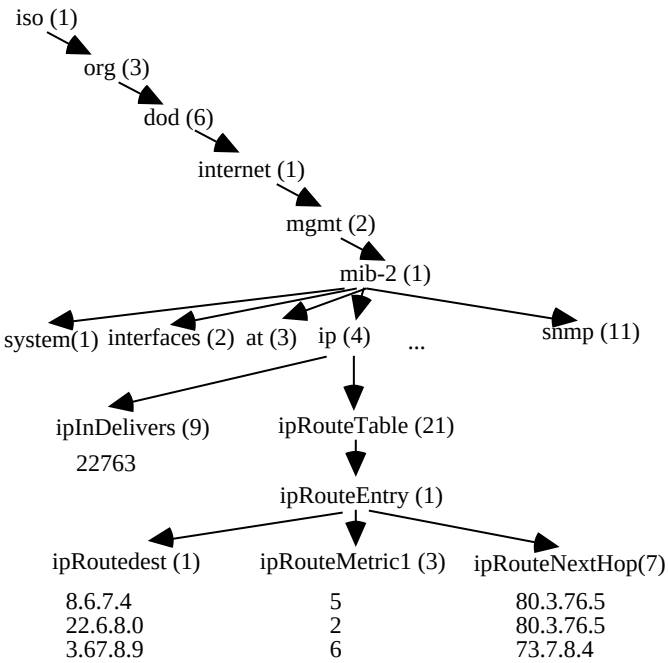
```
RFC1157-SNMP DEFINITIONS ::= BEGIN
IMPORTS
        ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
                FROM RFC1155-SMI;
Message ::= SEQUENCE {
                version INTEGER  {version-1 (0)},  -- Version 1 for this RFC
                community OCTET STRING,                 -- Community name
                data ANY                                -- e.g. PDUs
                        }
PDUs ::= CHOICE {  -- Protocol data units
                get-request             GetRequest-PDU,
                get-next-request        GetNextRequest-PDU,
                get-response            GetResponse-PDU,
                set-request             SetRequest-PDU,
                trap                    Trap-PDU
                }
GetRequest-PDU      ::= [0] IMPLICIT PDU
GetNextRequest-PDU::=[1] IMPLICIT PDU
GetResponse-PDU     ::= [2] IMPLICIT PDU
SetRequest-PDU      ::= [3] IMPLICIT PDU
PDU ::= SEQUENCE {
                request-id INTEGER,          -- Request identifier
                error-status INTEGER {               -- Sometimes ignored
                        noError (0),
                        toobig (1),
                        noSuchName (2),
                        badValue (3),
                        readOnly (4),
                        genError (5)},
                error-index INTEGER,                 -- Sometimes ignored
                                                     -- Index of the faulty variable
                variable-binding VarBindList }       -- Values are sometimes ignored
Trap-PDU ::= [4] IMPLICIT SEQUENCE {
                enterprise      OBJECT IDENTIFIER,      -- Type of object generating trap
                agent-addr      NetworkAddress,         -- Only one type of network adresses
                                                        -- IP adress of object generating trap
                generic-trap    INTEGER {               -- Generic trap type
                        coldStart (0),
                        warmStart ( 1),
                        linkDown ( 2),
                        linkUp (3),
                        authenticationFailure (4),
                        egpNeighborLoss ( 5),
                        enterpriseSpecific (6)
                        },
                specific-trap   INTEGER,   -- Specific code
                time-stamp TimeTicks,    -- Elapse time since the last reinitialization of the entity
                variable-binding VarBindList  -- "Interesting" information
                }
VarBind ::= SEQUENCE      - Variable binding
                {name ObjectName,
```

value ObjectSyntax}
VarBindList ::= SEQUENCE OF VarBind
END

La désignation des objets d'administration SNMP se fait selon une méthode de construction arborescente des noms où chaque nœud peut être repéré par une chaîne de caractères ou un code numérique entier. Pour décrire un nœud de l'arbre on utilise une notation pointée pour les noms logiques de l'Internet.

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

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



La désignation absolue de l'objet ipInDelivers est:

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

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

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

(ipRouteMetric1.22.6.8.0="2"),
(ipRouteNextHop 22.6.8.0="80.3.76.5"))

On continue la consultation de la table par :

GetNextRequest (ipRoutedest.22.6.8.0, ipRouteMetric1.22.6.8.0, ipRouteNextHop.22.6.8.0)

On obtient la réponse (l'entrée suivante de la table):

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

Enfin, par la dernière requête de la consultation de la table on obtient la dernière réponse qui correspond à la première ligne de la table:

GetNextRequest (ipRoutedest.3.67.8.9, ipRouteMetric1.3.67.8.9, ipRouteNextHop.3.67.8.9)

on obtient la réponse suivante :

GetResponse ((ipRoutedest.8.6.7.4="8.6.7.4"),
(ipRouteMetric1.8.6.7.4="5")
(ipRouteNextHop.8.6.7.4="80.3.76.5"))

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

I Questions ASN1

I.1. La spécification du message GetRequest-PDU comporte une directive **[0] IMPLICIT**. Que signifie-t-elle?

I.2. Dans la spécification ASN.1 dont vous disposez, quel est le type qui décrit les objets administrés par les agents SNMP ?

I.3. A quoi correspondent les différents champs du type PDU?

I.4. A la vue des commentaires de la description du type PDU, réécrivez ce type en ASN.1 en cherchant à optimiser l'encombrement des messages (certains champs ne sont pas toujours obligatoires) ? Justifiez votre réécriture

II Question d'organisation architecturale

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

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

III Questions SNMP

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

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

IV Questions de spécifications d'applications

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

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

Exercice 9 : Client serveur graphique : Protocole X

Le protocole X utilisé par le logiciel de multifenêtrage X Window permet de construire un système graphique de sorte qu'une application graphique peut s'exécuter sur un site et que son affichage se fasse néanmoins sur un autre site.

La solution est d'utiliser un fonctionnement sous forme client/serveur. Le programme A envoie des demandes (requêtes) de la forme "afficher moi une droite SVP". Le programme X reçoit ces demandes et les exécute si possible. Le site d'exécution du programme X est celui où l'utilisateur voit s'afficher les graphismes demandés.

1) Dans un tel système lequel des programmes A et X est un client graphique ? Lequel est le serveur graphique ? Si une personne travaillant devant un terminal ou une station de travail fait exécuter un programme sur une machine distante et l'affiche sous ses yeux, sur quelle machine locale ou distante est exécutée le serveur graphique ? En quoi cette solution est étonnante quant à la localisation des clients et des serveurs par rapport à celle réalisée par exemple dans un service de fichiers comme NFS ?

2) En utilisant quel type de service de la hiérarchie OSI un tel protocole doit il être implémenté? Si l'on veut construire le protocole facilement, ce service doit il être fiable ou non, permettre le déséquencement et la perte de message ou non ?

3) À quel niveau de l'architecture ISO-OSI un tel protocole se place t-il ?

4) Plusieurs applications graphiques distantes peuvent afficher des fenêtres différentes sur un même écran graphique géré par le serveur graphique. Indiquer alors les premiers travaux que doit faire un serveur graphique relativement aux messages reçus et aux directives graphiques données par l'usager (souris, ...).

5) Le protocole graphique adopté dans ce problème est le protocole X défini par le MIT. Historiquement il a été appuyé par plusieurs constructeur (IBM, DEC, ...). Indiquer ce que doit définir un tel protocole en termes d'ouverture.

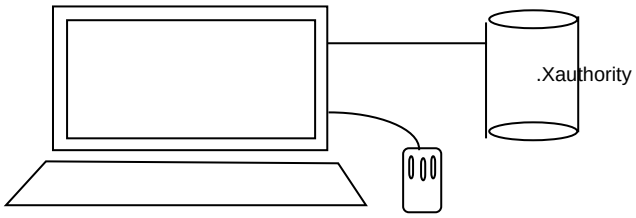
6) Pour la suite du problème, on parlera de serveur X au lieu de serveur graphique et de client X au lieu de client graphique. Entre un client X et un serveur X des messages sont véhiculés. Ces messages sont dans un sens appelés des requêtes et dans l'autre sens des événements. Indiquer le nom des messages dans le sens client -> serveur et dans le sens serveur -> client. Un message comme "afficher moi une droite SVP" est il une requête ou un événement ? Même question pour un message comme "On a appuyé sur le bouton gauche de la souris alors qu'on était dans une fenêtre menu ".

7) Dans les premières versions n'importe quel utilisateur U1 connecté sur une machine M1 pouvait lancer une application en indiquant que l'affichage se fait sur une machine M2 (et l'écran de celle-ci). Indiquer les inconvénients d'un tel procédé.

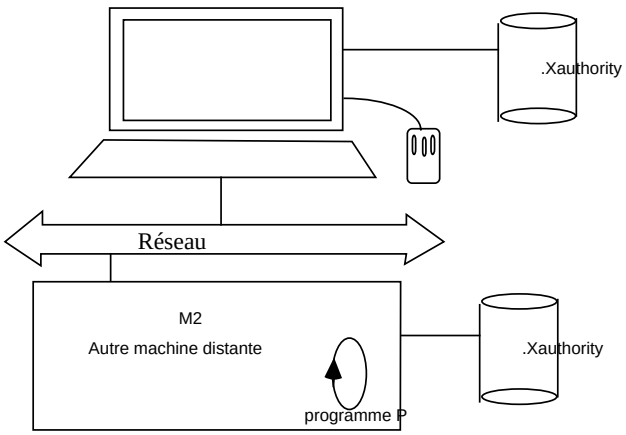
8) On a proposé une première solution consistant à n'autoriser l'accès à un serveur X qu'à certaines machines. Une machine M1 possède un fichier personnel (/etc/X0.hosts) dans lequel est spécifié une liste de machine M'1, M'2, M'3 par exemple. Seules les applications exécutées sur ces machines M'1, M'2, M'3 peuvent afficher sur la machine M1. Indiquer comment un utilisateur mal intentionné peut afficher sur la machine M1.

9) Pour éviter ce problème, chaque utilisateur lorsqu'il lance le système de fenêtrages X11 sur la machine M1 reçoit un numéro aléatoire (appelé MIT-MAGIC-COOKIE-1). Ce numéro est placé dans un fichier personnel (le .Xauthority) et est connu du serveur X gérant la machine M1. On a donc dans ce fichier personnel de l'utilisateur une ligne comme :
M1:26785.
Chaque fois qu'une application A est lancée par l'utilisateur U1, cette application communique au serveur X gérant la machine M1 ce numéro. Expliquez pourquoi un utilisateur U2 ne peut plus, pendant l'utilisation de la machine M1 par U1, afficher sur M1.
Remarque : les lignes M1:num ne sont pas effacées du fichier personnel mais réécrites à chaque nouvelle connexion au serveur X de la machine.

10) Plus précisément un utilisateur U est connecté sur une machine M1 sur laquelle il a lancé une session X Window. Le serveur X de cette machine a donc mis dans le fichier .Xauthority de l'utilisateur la ligne :
M1:26785
et il connaît cette valeur. Cette machine est une station de travail qui est capable non seulement d'exécuter un serveur X mais aussi d'exécuter des programmes graphiques. Chaque fois que U veut exécuter un programme graphique sur M1 (pour l'afficher sur M1), ce programme va lire le fichier .Xauthority qui fournit la valeur 26785 et la passe au serveur X. Celui-ci reconnaît la valeur et accepte l'affichage du programme. L'ensemble des éléments intervenant dans ce scénario sont locaux à la machine M1.



Lorsque l'utilisateur U tout en restant dans la même session X Window se connecte sur une machine distante M2 et demande à exécuter un programme graphique P sur la machine M2 avec affichage sur la machine M1, ce programme va lire le fichier .Xauthority de l'utilisateur U sur la machine M2.



Ce fichier .Xauthority de la machine M2 est a priori distinct du fichier .Xauthority de la machine M1 même s'ils concernent tous deux le même utilisateur U. Il n'a donc pas de raison de contenir la ligne M1:26785. Le programme P ne peut donc pas proposer le bon numéro de session 26785 au serveur X de la machine M1 et va être rejeté.
11) Quelle solution proposez-vous pour éviter ce problème. On pourra penser à utiliser NFS.
12) Avec une telle solution, quel(s) problème(s) en terme de sécurité réseau subsiste(nt)?

Exercice 10 : 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 11 : XML

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

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

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

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

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

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

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

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

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

</BIBLIOGRAPHIE>
```

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

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

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

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

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

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

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

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

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

Exercice 12 : 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.

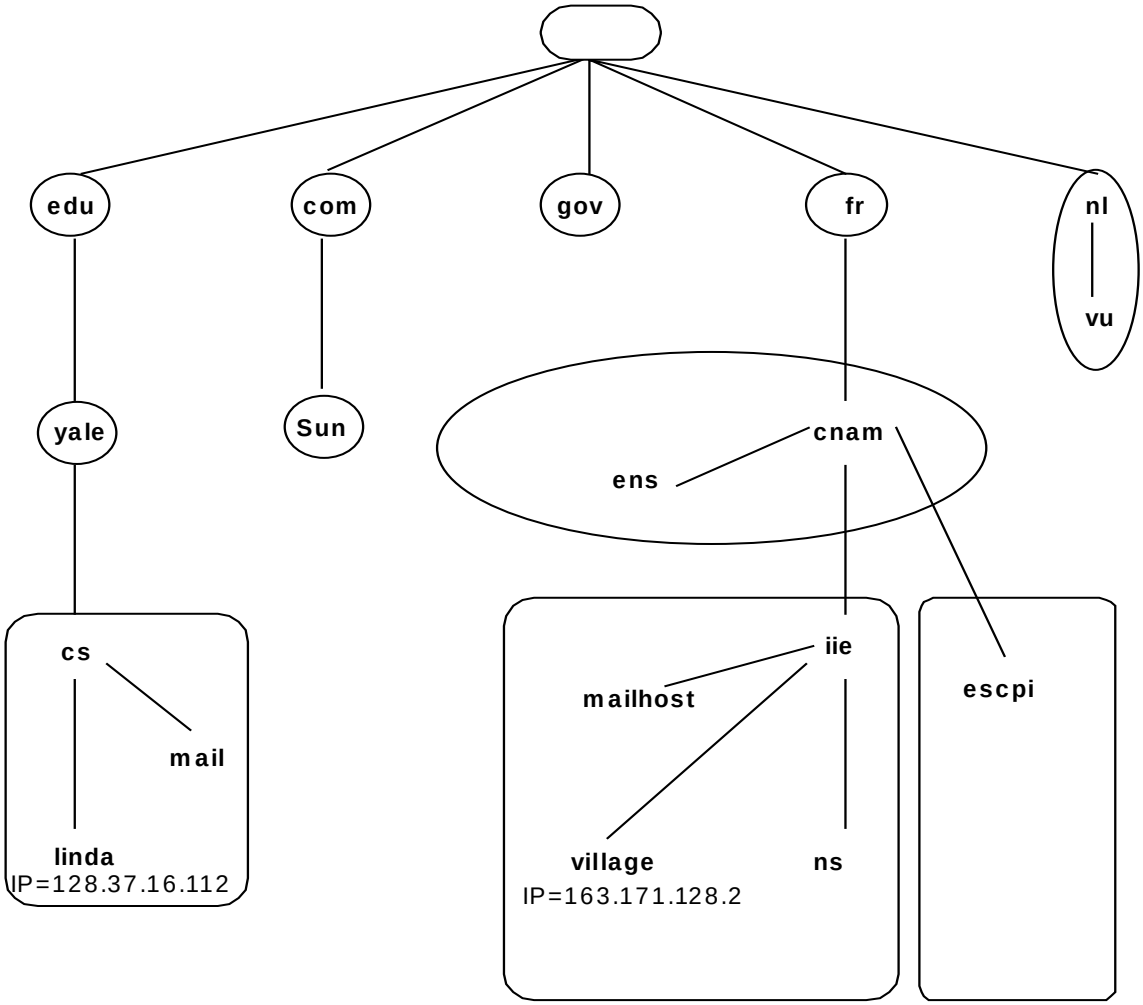
12) L'utilisateur a modifié la configuration de son client navigateur de manière à ne plus utiliser de cache local client mais un cache serveur sur son site.

12.1) Comment se comporte maintenant un client?

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

Exercice 13 : 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'usager asterix (le nom le plus probable sur le dessin)?

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

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

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

Nom_de_domaine, Durée_de_vie, Type, Classe, Valeur

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

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

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

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

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

6 En pratique, les serveurs DNS envoient des messages qui contiennent plusieurs requêtes ou plusieurs réponses.
Quel est le nom de cette technique? Quel est son utilité ?

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

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

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

9 On améliore l'apprentissage des résolutions de requêtes par une seconde technique de résolution baptisée résolution récursive. Contrairement à la résolution itérative la réponse fait le chemin inverse de la requête et est donc apprise par tous les sites traversés. Dans le cadre de cette dernière solution, lorsque superman@ linda.cs.yale.edu envoie un courrier à asterix@village.iie.cnam.fr, quelle est la liste des serveurs consultés et quel est le parcours des messages ?

Exercice 14 : 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
6.17.173.163.in-addr.arpa
ulysse.cnam.fr
Quel est le motif le plus probable qui conduit à la création de ces noms de domaine?

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

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

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

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

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 15 : 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 autre 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.

1.2.4 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.

1.2.5 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.

1.2.6 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 16 : Messagerie Internet SMTP

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

I Architecture de la messagerie SMTP

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

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

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

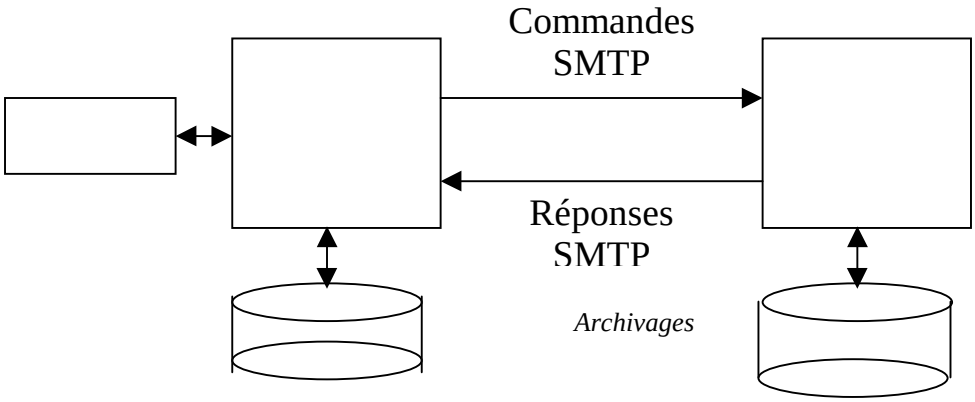
I.4) Il existe deux grandes catégories d'architectures pour les messageries. La première catégorie est basée sur un principe de délivrance de bout en bout ('end to end delivery'). La seconde catégorie est basée sur une approche de stockage et retransmission ('store and forward delivery'). Rappelez les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

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

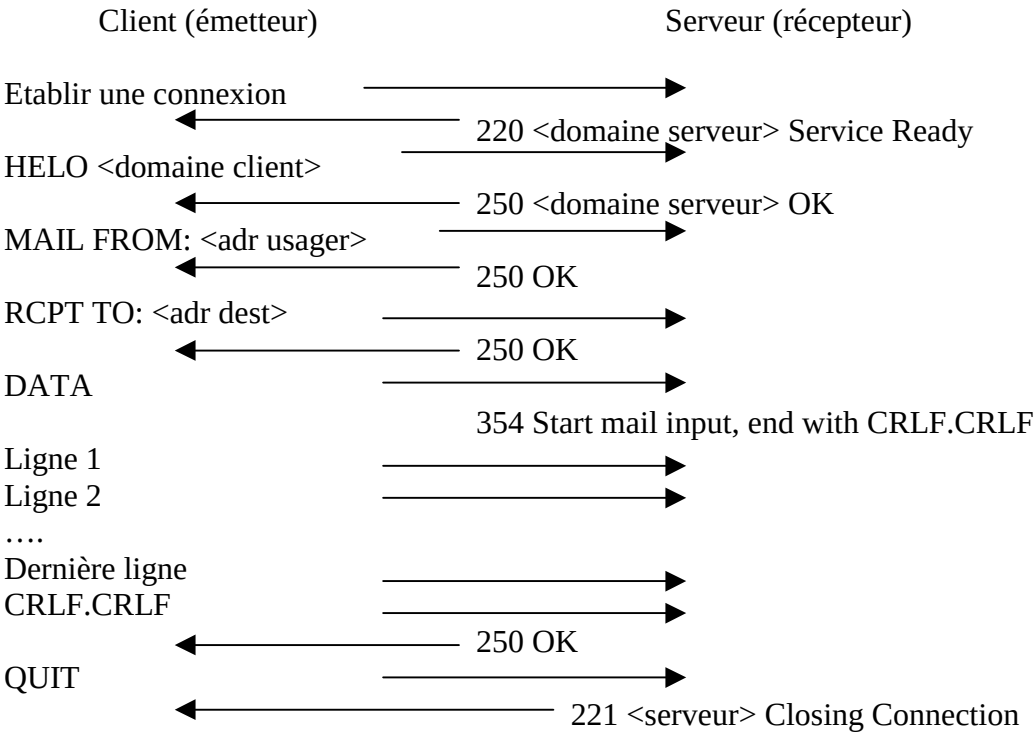
I.6) Quelles sont les différentes étapes de la transmission d'un courrier électronique dans le cas habituel d'un prestataire de service d'accès Internet ou d'une entreprise de taille significative. Précisez les matériels (ordinateurs), les logiciels (agent utilisateur, serveur de messagerie, donnez des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

II Le protocole d'émission de courrier SMTP

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



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



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

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

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

telnet cnam.fr 25

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

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

III Protocoles de réception de courrier

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

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

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

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

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

La phase d'authentification ('Authorization' en anglais)

La phase de transaction ('Transaction' en anglais)

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

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

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

server_response::= simpleresponse /
                        statresponse/
                        errorresponse

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

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

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

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

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

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

Le contenu du message est le suivant:

```
X-Mailer: Microsoft Outlook Express Macintosh Edition - 4.5
(0410)
Date: Sat, 02 Jun 2001 15:26:38 +0200
Subject: Publi
From: "gerard" <gerard@cnam.fr >
To: natkin@cnam.fr
Mime-version: 1.0
X-Priority: 3
Content-type: multipart/mixed;
   boundary="MS_Mac_OE_3074340399_15662732_MIME_Part"

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

Ci joint les publications.

--
G=E9rard Florin
```

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

0M8R4KGxGuEAAAAAAAAAAAAAAAAAAAAPgADAP7/CQAGAAAAAAAAAAAAAAABAA
AARQAAAA
AAAAEAAARwAAAAEAAAD+////AAAAAEQAAAD/////////////////////////
////////

--MS_Mac_OE_3074340399_15662732_MIME_Part--
```

III.4) Donner pour chaque message échangé (chaque PDU échangée) dans l'exemple d'échange précédent la phase à laquelle il appartient?

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

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

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

III.8) Expliquez ce que signifient

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

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

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

Représentez l'automate possédant les cinq états.

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

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

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

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