

CONCOURS DE RECRUTEMENT DES TECHNOLOGUES EN INFORMATIQUE SESSION 2002

Admissibilité

Epreuve d'application Option : Informatique des systèmes industriels

GESTION INFORMATISEE DU TRAFIC AERIEN CIVIL

A/ PRESENTATION

1 - Description générale

L'étude proposée porte sur l'informatisation de la gestion du trafic aérien civil. Le système¹ (cf. figure 1) est composé de :

- Trois stations radar de surveillance réparties sur le territoire : Tunis-Carthage, Sidi-Zid et Akouda.
- Le centre de la navigation aérienne situé à Tunis.
- Les différents postes de contrôle situés aux aéroports.

Chaque radar a une zone de couverture qui s'étend sur un rayon approximatif de 400 Km. Il balaie en permanence sa zone de couverture et transmet par voie Hertzienne et en temps réel les informations collectées au centre de la navigation aérienne situé à Tunis. Le radar met 4,6s pour effectuer un balayage de toute sa zone.

¹ Bien qu'inspiré de la réalité, le système décrit comporte plusieurs simplifications pour les besoins du sujet.

Le centre de navigation aérienne comprend deux sous systèmes implantés chacun sur un ordinateur :

- Le sous système de gestion de l'espace aérien (SSGEA) qui traite les informations recueillies par les radars et reconstitue une vue globale de l'espace aérien.
- Le sous système de gestion des plans de vol (SSGPV) : Les informations relatives aux plans de vol des avions proviennent du réseau des services fixes de télécommunications aéronautiques. Le SSGPV s'interface à ce réseau pour maintenir une vue locale de cette base de données.

De plus, le centre de navigation aérienne est relié par des lignes spécialisées aux différents centres de contrôle aérien. Chaque centre de contrôle héberge plusieurs postes identiques permettant aux contrôleurs aériens de suivre en temps réel le trafic et d'intervenir pour le gérer. Chaque poste de contrôle sera désigné dans la suite par le sous système local de visualisation (SSLV).

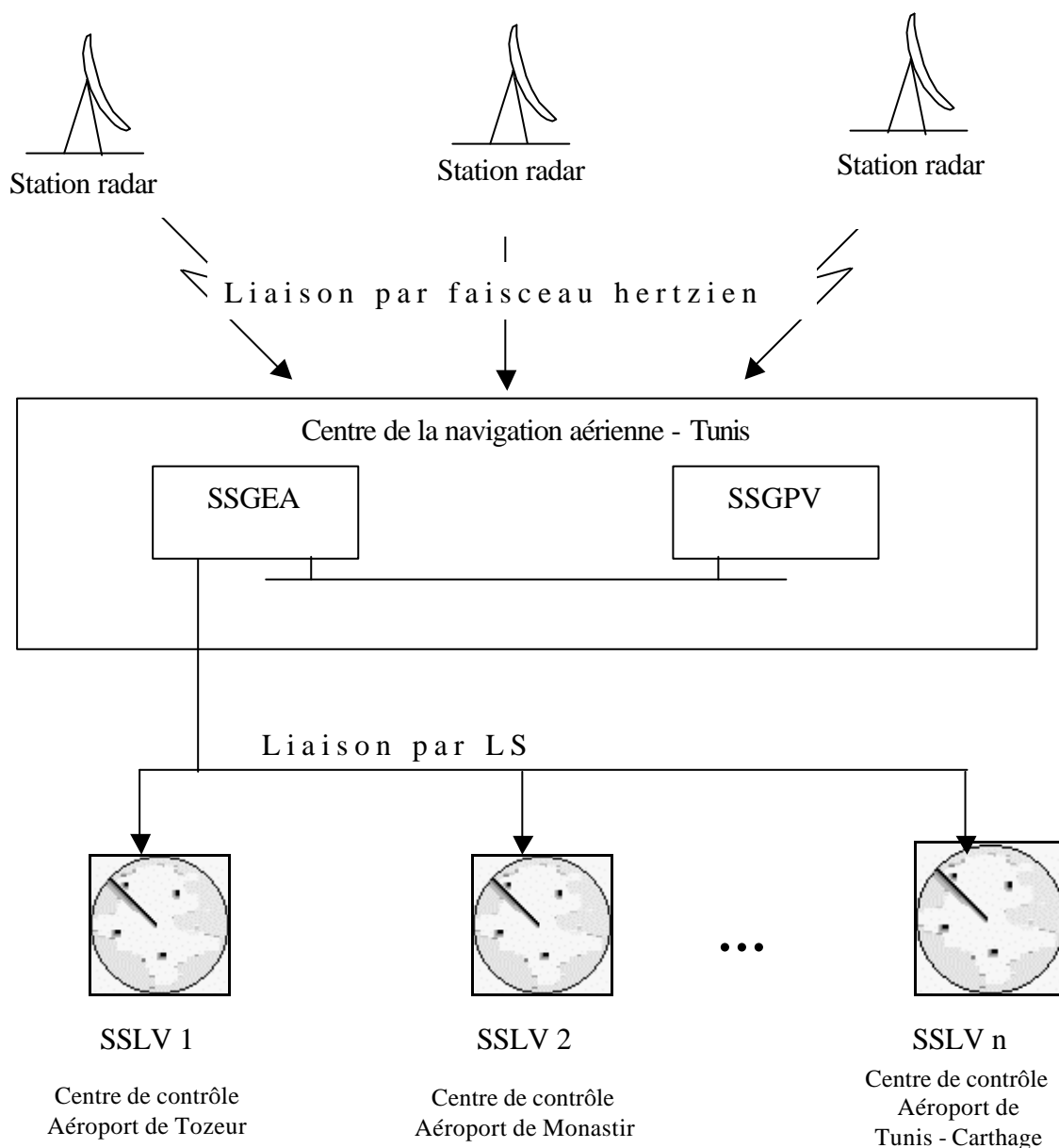


Figure 1 – Architecture générale du système de Gestion de la navigation aérienne

1.1 - Principes généraux du contrôle aérien

Les avions qui traversent l'espace aérien sont pour certains équipés d'un transpondeur et pour d'autres non. Le transpondeur est constitué d'un émetteur/récepteur en mesure de fournir l'identification de l'avion. Quand un avion rentre dans la zone de couverture de l'un des radars, ce dernier le détecte et calcule sa position en coordonnées cylindriques (, ,h) par rapport à son propre repère. Ensuite il interroge son transpondeur. Lorsqu'il obtient une réponse il transmet au SSGEA une trame d'information contenant le code transpondeur de l'avion ainsi que sa position sinon il ne transmet que la position de l'avion détecté mais non reconnu.

Chaque station radar possède dans sa proximité un transpondeur fixe (appelé CPME) dont elle connaît les coordonnées exactes. A chaque balayage, le radar détecte son CPME et calcule sa position comme tout avion possédant un transpondeur. Le SSGEA, lorsqu'il reçoit la trame correspondante, doit se rendre compte qu'il s'agit du CPME et connaissant la position de ce dernier il peut calculer un facteur de correction correspondant au décalage entre position détectée et position connue. Ce facteur sera utilisé pour corriger toutes les prochaines trames reçues.

1.2 - Caractéristiques d'un vol

Un vol est caractérisé par :

- Son indicatif dit aussi N° de vol (Par exemple TU220) qui permet de l'identifier de façon unique.
- Ses villes de départ, d'escales éventuelles et d'arrivée.
- La route aérienne qu'il lui a été affectée. Il est à noter que la route n'est pas définitive, elle peut être modifiée par le contrôleur aérien à tout moment en fonction de l'encombrement de l'espace aérien, des conditions climatiques, etc. En particulier, dans le cas où il s'agit d'un avion non reconnu, le contrôleur lui affecte une route.

Notons que tout comme la circulation terrestre, la circulation aérienne emprunte des routes aériennes bien définies. Il peut exister plusieurs routes entre deux aéroports.

Toutes les informations relatives à tous les vols internationaux et nationaux sont regroupées au sein du système d'information géré par le réseau des services fixes de télécommunications aéronautiques (cf. §1).

1.3 - Responsabilités des contrôleurs

Dès qu'un avion pénètre l'espace aérien sous surveillance, il est pris en charge par l'un des contrôleurs aériens qui a la responsabilité de s'assurer que :

- L'avion ne dévie pas au delà d'une certaine marge de la route qu'il lui a été affectée. Le contrôleur a alors la responsabilité de contacter le pilote pour lui demander de corriger le cap.
- L'avion ne descend pas en dessous d'une altitude minimale lorsqu'il est en phase de croisière. Le contrôleur a alors la responsabilité d'aviser le pilote pour lui demander de remonter.

- Deux avions ne se rapprochent pas l'un de l'autre au delà d'un rayon de sécurité de 5Km. Cette distance de sécurité permet au contrôleur de réagir et de rétablir la situation avant une collision.
- L'avion ne disparaît pas alors qu'il est encore dans l'espace sous contrôle. Lorsque cela se produit, le contrôleur a la responsabilité de déclencher une procédure d'urgence.

Le système informatisé de la navigation aérienne devra assister les contrôleurs aériens en fournissant une image en temps réel de l'espace aérien surveillé, en détectant et en signalant, le cas échéant, les situations d'alarme.

L'étude portera sur la spécification et la conception du sous système de gestion de l'espace aérien, du sous système de gestion des plans de vol, du sous système local de visualisation et de la communication entre les différents sous systèmes.

2 - Etude du sous système de la gestion de l'espace aérien (SSGEA)

Le SSGEA est implanté sur un PC équipé d'une carte multi-canaux qui reçoit les trames d'information envoyées par les stations radar sur les ondes hertziennes.

A chaque balayage et pour chaque avion détecté dans sa zone de couverture, chaque station radar envoie au SSGEA une trame d'information contenant le code transpondeur de l'avion, s'il en possède un, et ses coordonnées cylindriques (\tilde{n}, \tilde{e}, h).

A la réception de cette trame, le SSGEA doit :

- Avant tout lui associer une date (estampille temporelle) qui correspond à l'instant de détection de l'avion à cette position.
- Il corrige éventuellement la position fournie par le radar en se référant aux coordonnées du CPME.
- Il ramène les coordonnées par rapport au repère commun. Les coordonnées reçues à partir de chaque station radar ont pour origine la station radar et pour plan de référence un plan vertical orienté vers le nord. Le SSGEA transforme toutes les coordonnées dans un repère cylindrique ayant pour origine le centre de la navigation aérienne et pour plan de référence le même plan vertical orienté vers le nord.
- Il doit identifier à quel avion appartient cette trame :
 - o si la trame contient un code transpondeur, ce dernier permet de reconnaître l'avion.
 - o si la trame ne contient pas un code transpondeur, le SSGEA doit déterminer l'avion auquel appartient cette trame en fonction des dernières positions détectées et en fonction de la vitesse constatée des différents avions.
- Le SSGEA envoie alors au SSLV un message de type PLANE_POSITION contenant l'identifiant de l'avion, sa position, sa vitesse et sa direction.
- Si c'est la première fois où l'avion est détecté, le SSGEA utilise le code transpondeur de l'avion pour interroger le SSGPV pour obtenir les différentes informations liées au

vol de l'avion et lui associe un identifiant qui sera utilisé pour les échanges avec les différents SSLV. Si le code transpondeur de ce dernier n'est pas connu il lui attribue un identifiant et le classe comme non reconnu.

- A la première détection d'un avion, le SSGEA envoie au SSLV un message du type PLANE_ENTER contenant l'identifiant de l'avion, une indication reconnu ou non et s'il est reconnu, son numéro de vol et sa position.
- Pour chaque avion détecté le SSGEA s'occupe également de vérifier l'occurrence de chacune des situations d'alarmes exposées plus haut. A la constatation d'une alarme il envoie au SSLV un message du type ALARME contenant le code de l'avion correspondant et le type d'alarme (E = Ecart par rapport à la route, H= Hauteur limite non respectée, R= Rapprochement d'un autre avion et D = Disparition).
- Quand l'avion n'est plus détecté pendant 2 balayages successifs alors qu'il est à la limite de l'espace aérien sous surveillance, il est considéré comme ayant quitté cet espace sous contrôle et le SSGEA envoie au SSLV un message du Type PLANE_LEFT contenant uniquement l'identifiant de l'avion.

Récapitulatif des trames échangées :

PLANE_ENTER : ID_Avion + Reconnu + N° du Vol + + + h

PLANE_POSITION : ID_Avion + Position_ + Position_ + Position_h + Vitesse +
Direction_ + Direction_ + Direction_h

ALARME : ID_Avion + Type_Alarme

PLANE_LEFT : ID_Avion

ROUTE_CHANGE : ID_Avion + ID_Route

3 - Etude du sous système local de visualisation (SSLV)

En permanence le SSGEA diffuse simultanément vers les différentes SSLV des messages. Ces derniers sont traités de manière identiques par les SSLV. De ce fait, les différents SSLV ont la même connaissance de l'espace aérien mais suivant les interactions avec les contrôleurs aériens, peuvent en présenter des vues différentes.

Les messages traités par le SSLV sont les suivantes :

- Entrée d'un avion: Lorsque le SSGEA détecte un nouvel avion il envoie une commande PLANE_ENTER. Les paramètres de cette commande sont :
 - o Un identifiant unique pour l'avion qui sera utilisé dans tous les échanges ultérieurs entre le SSGEA et le SSLV pour référencer cet avion.
 - o La nature de cet avion: Reconnu ou non reconnu. Un avion reconnu est un avion qui dispose d'un code transpondeur. Le SSLV représente par des icônes différentes les avions reconnus et non reconnus.
 - o Dans le cas d'un avion reconnu, le SSGEA fournit aussi l'indicatif de l'avion.
 - o La position de l'avion, sa vitesse et sa direction.

- Sortie d'un avion: Lorsque un avion quitte l'espace aérien surveillé, le SSGEA émet vers le SSLV une commande PLANE_LEFT avec comme paramètre unique l'identifiant de l'avion. Il est à noter que le SSGEA pourra maintenant exploiter l'identifiant pour désigner un nouvel avion.
- Position d'un avion: A chaque balayage de l'espace aérien par le radar, le SSGEA calcule pour chaque avion détecté sa position par rapport à un repère commun. Il transmet une commande PLANE_POSITION qui a deux paramètres :
 - o L'identifiant de l'avion
 - o Sa position sous la forme de coordonnées \tilde{n}, \tilde{e} et h par rapport à un repère cylindrique fixe ainsi que sa vitesse et sa direction.
- Alarmes: Lorsqu'une situation d'alarme se produit, le SSGEA transmet vers le SSLV un message ALARME qui a deux paramètres :
 - o La nature de l'alarme: Route, Altitude, Collision, Disparition
 - o L'identifiant de l'avion concerné.

De plus, le SSLV peut transmettre vers le SSGEA un message ROUTE_CHANGE dont les paramètres sont :

- L'identifiant de l'avion.
- Un numéro de route.

Ce message permet de signaler au SSGEA qu'un avion a été dérouté par un contrôleur aérien.

Principales fonctions du SSLV

Le SSLV offre cinq fonctions majeures :

1. L'écoute de messages: afin de pouvoir traiter les messages en provenance du SSGEA, le SSLV doit offrir une fonction d'écoute asynchrone qui s'interface avec le système de communication. Celle-ci devra en permanence recueillir les messages qui arrivent sur le réseau, les décoder et les placer dans une file de messages.
2. Le traitement de commandes: le contrôleur aérien interagit avec le SSLV au travers d'un clavier qui lui permet d'activer trois fonctions :
 - Définition du zoom: le contrôleur peut visualiser l'espace à différents niveaux de zoom. En effet, il existe 6 niveaux de zoom (50%, 100%, 150%, 200%, 300% et 500%). L'opérateur peut changer de niveau à tout moment. Suite à un changement de zoom le SSLV doit déterminer les éléments graphiques ou textuels qui continuent de s'afficher et procéder à leur réaffichage avec la nouvelle échelle.
 - Définition de la fenêtre de visualisation: la taille de l'écran du contrôleur ne permet pas toujours de visualiser l'ensemble de l'espace aérien, en particulier lorsque le zoom est élevé, seule une partie de l'espace peut être représentée. Le contrôleur a la possibilité de déplacer la vue pour observer la partie de l'espace qui l'intéresse.

- La modification de route : lorsqu'il le juge nécessaire, le contrôleur peut décider de dérouter un avion auquel cas il le signale au système afin de ne pas recevoir de messages d'alarmes concernant la route de cet avion. Le SSLV devra transmettre au SSGEA un message ROUTE_CHANGE. Pour cela il devra créer le message correspondant et le placer dans une file de messages en attente d'être transmis.
3. L'émission de messages : afin de pouvoir émettre les messages de changement de route, le SSLV doit offrir une fonction d'émission asynchrone qui s'interface avec le système de communication. Celle-ci devra en permanence recueillir les demandes de messages en attente d'être transmis, construire les messages correspondant et s'interfacer avec le système de communication pour les émettre.
 4. La visualisation de l'espace aérien : le SSLV affiche en permanence un fond constitué par l'ensemble des routes aériennes de l'espace surveillé et le trafic aérien sous forme d'avions et de trajectoires d'avions. Aussi bien les routes que les trajectoires des avions sont constituées par des segments. Le SSLV réagit aux messages émis par le SSGEA et aux commandes des opérateurs en assurant les fonctions suivantes :
 - Création d'un avion : le SSLV devra mettre à jour ses structures de données pour mémoriser l'entrée d'un avion dans l'espace aérien, il devra aussi mémoriser toutes les caractéristiques de l'avion. Il devra aussi éventuellement afficher à l'écran l'icône de l'avion et son étiquette.
 - Déplacement d'un avion : le SSLV devra mettre à jour la trajectoire suivie par l'avion et mettre à jour éventuellement les affichages.
 - Suppression d'un avion : le SSLV devra transférer sur un stockage permanent toutes les informations concernant l'avion et la trajectoire qu'il a suivie puis libérer les structures de données qui lui étaient réservées et mettre à jour éventuellement l'affichage.
 - Déplacement de la vue : le SSLV devra mémoriser les coordonnées de la région visible et mettre à jour les affichages.
 - Modification de zoom : le SSLV devra mémoriser le nouveau zoom et mettre à jour les affichages.
 5. La visualisation des alarmes : A chaque occurrence de message d'alarmes, le SSLV devra dessiner en rouge et faire clignoter à l'écran l'avion concerné par l'alarme et en même temps afficher un message d'alarme approprié.

La figure 2 donne la représentation sous forme de diagramme de cas d'utilisations, dans le formalisme d'UML (Cf. Annexe A3 pour les éléments de la notation UML)

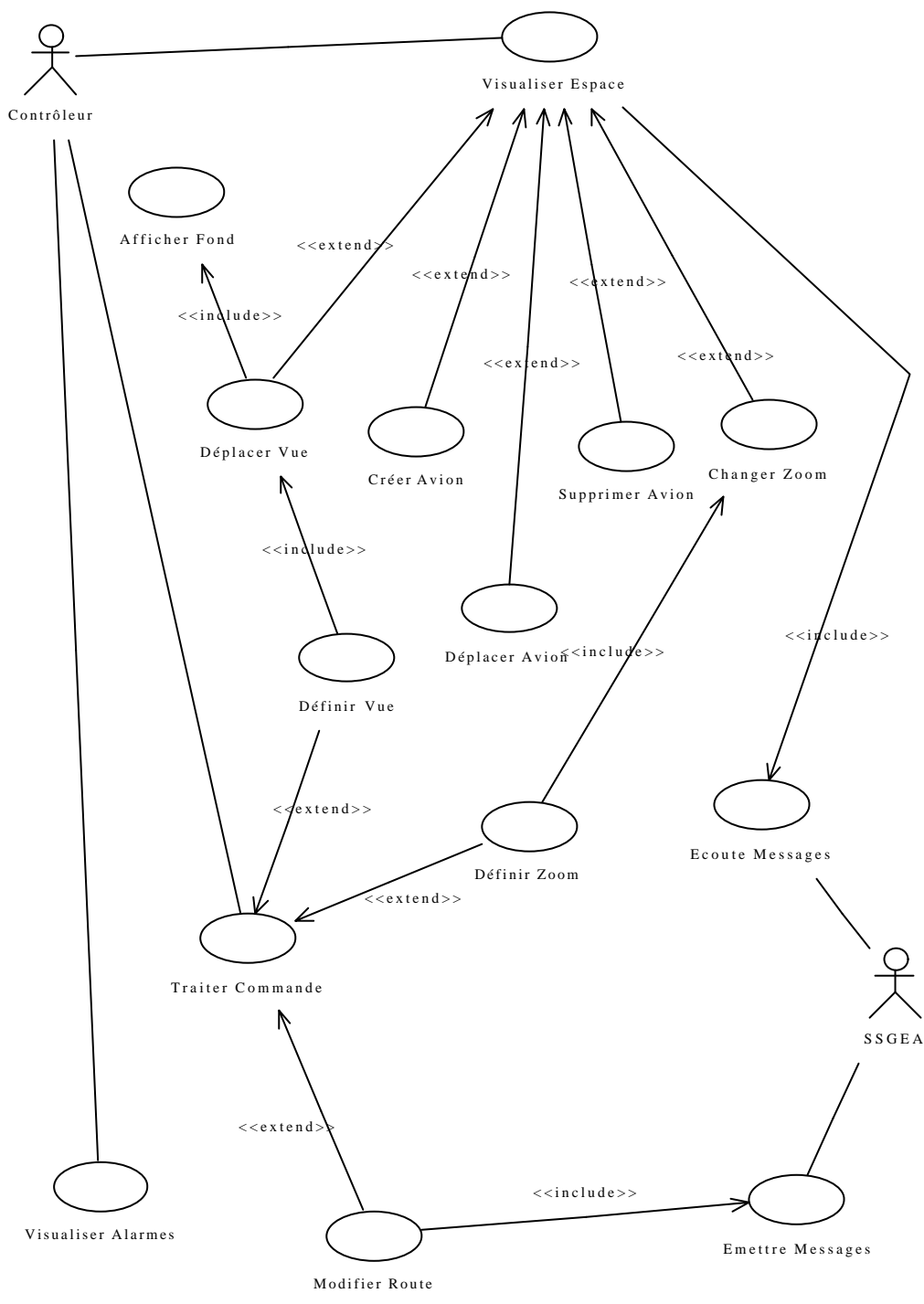


Figure 2 : Diagramme des cas d'utilisations pour le SSLV

Le diagramme des cas d'utilisations permet d'identifier les objets suivants :

- **Affichable** : tous les éléments qui s'affichent sur l'écran de l'opérateur soit sous forme graphique (comme par exemple les icônes des avions) soit sous forme graphique et textuelle (comme les messages d'alarmes) constituent des objets affichables.
- **Alarme** : c'est l'objet qui matérialise l'une des trois alarmes reçues par le SSLV.
- **Avion** : tous les avions qui passent par l'espace aérien correspondent à des objets avions.
- **Commande** : toute interaction a travers le clavier faite par l'opérateur aboutit à la création d'un objet commande.
- **Commande_Fenetre** : cet objet matérialise une commande de positionnement de la fenêtre de visualisation.
- **Commande_Route** : cet objet matérialise une commande de changement route.
- **Commande_Zoom** : cet objet matérialise une commande de changement du zoom.
- **Controleur_Clavier** : à tout moment il n'y a qu'un seul exemplaire de cet objet qui se charge de l'interfaçage avec le clavier et qui détecte et construit toutes les objets correspondants aux commandes de l'opérateur.
- **Contrôleur_Ecran** : à tout moment il n'y a qu'un seul exemplaire de cet objet qui se charge de gérer tous les affichages.
- **Creation_Avion** : cet objet est créé suite à l'arrivée d'un message PLANE_ENTER
- **Deplacement_Avion** : cet objet est créé suite à l'arrivée d'un message PLANE_POSITION.
- **Ecran** : à tout moment il n'y a qu'un seul exemplaire de cet objet qui s'interface avec le dispositif matériel constituant l'écran et qui permet de mémoriser les éléments en cours d'affichage à un instant donné.
- **Etiquette** : chaque avion s'affiche à l'écran sous forme d'une icône accompagnée d'une étiquette, l'étiquette donne les informations pertinentes pour l'avion (son indicatif, son altitude, sa vitesse,...).
- **Fond** : le fond de l'écran est composé de l'ensemble des routes que peuvent suivre les avions.
- **Icône** : les icônes sont la représentation graphique des avions.
- **Icône_Avion Identifié** : il s'agit de la représentation graphique d'un avion identifié.
- **Icône_Avion_Inconnu** : il s'agit de la représentation graphique d'un avion non identifié.
- **In_Message_Queue** : cet objet existe aussi sous forme d'exemplaire unique, il permet de gérer la file des messages qui arrivent au SSLV.
- **Listener** : cet objet existe aussi sous forme d'exemplaire unique, il s'interface avec le réseau et reste en écoute perpétuelle, c'est lui qui détecte les messages entrants au SSLV et qui construit les objets messages correspondants et les place dans le file In_Message_Queue.
- **Message** : représente les messages qui arrivent au SSLV.

- Out_Commandes_Queue : cet objet existe aussi sous forme d'exemplaire unique, il permet de gérer la file des commandes qui sont émises par le SSLV.
- Point : Il s'agit d'un point de l'écran.
- Route : Il s'agit d'une des routes que doivent suivre les avions. Une route est composée de segments. Pour des raisons de sécurité évidentes il n'y a jamais d'intersection entre les routes.
- Segment : un segment est défini par deux points. Il s'affiche à l'écran par une segment de droite.
- Suppression_Avion : cet objet est créé suite à l'arrivée d'un message PLANE_LEFT.
- Transmitter : cet objet existe aussi sous forme d'exemplaire unique, il s'interface avec le réseau et scrute régulièrement la file Out_Commandes_Queue, dès qu'il y a des commandes dans cette file il se charge de les transmettre.

4 - Etude de la communication entre SSGEA et SSLV

La communication sera assurée (au niveau matériel) par un réseau privé entre le SSGEA et les différents SSLV réalisés par des LS (lignes spécialisées). Notons que le réseau est dimensionné de sorte qu'il réponde aux exigences de l'application.

Dans la partie étude du sous système SSLV §3, on a dégagé les objets «Listener» et «Transmitter» qui seront implémentés sous l'environnement UNIX. En effet, l'implémentation de ces objets sera faite entre autres par des processus UNIX (voir figure 3). L'échange entre ses différents processus communicants se fera selon le protocole TCP/IP. Ces échanges sont bidirectionnels :

- Dans le premier sens SSGEA \rightleftarrows SSLV : les messages diffusés.
- Dans le deuxième sens SSLV \rightleftarrows SSGEA : la modification de route

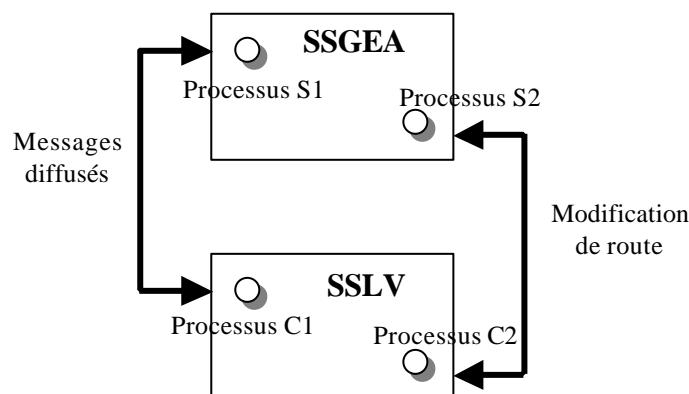


Figure 3 – Schéma de communication entre le SSGEA et un SSLV

On remarquera sur le schéma suivant (voir figure 4) le principe de communication pour un processus UNIX faisant partie d'un schéma client-serveur en utilisant les sockets UNIX. L'annexe A4 détaille l'utilisation des primitives de la bibliothèque Sockets sous UNIX.

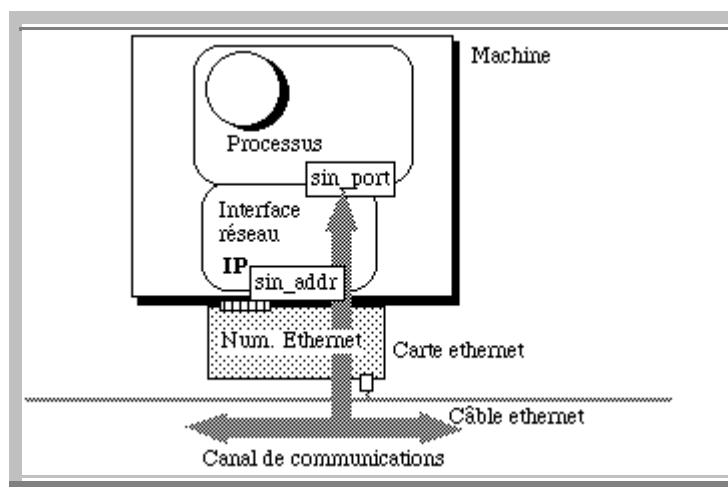


Figure 4 – Principe de la communication via une socket UNIX

- `sin_port`, le numéro de port,
- `sin_addr`, le(s) numéro(s) InterNet de la machine, numéro attribué par l'administrateur système (un par carte Ethernet sur la machine)
- le numéro Ethernet de la carte, numéro unique attribué (matériel) par le constructeur,

C'est sur ce type de canevas que l'on construit de nombreuses applications distribuées.

Ici, le serveur se contente d'attendre les demandes des clients distants. Dès qu'un client demande une connexion, la gestion de la communication avec ce client sera prise en charge par un processus fils, pendant que le serveur se remet en attente de demande de connexions d'autres clients.

5 - Etude du sous système de la gestion des plans de vol (SSGPV)

L'ensemble des informations concernant les vols nationaux et internationaux est géré par le réseau fixe des télécommunications aéronautiques. Celui-ci maintient une base de données structurant les données pertinentes sur les déplacements aériens. Toute organisme qui le souhaite peut s'interfacer à ce système pour obtenir les informations qui le concernent.

Le SSGPV a donc pour rôle de gérer une vue locale des informations nécessaires à la gestion informatisée du trafic aérien civil. Cette vue est régulièrement alimentée en données à partir du réseau fixe des télécommunications aéronautiques. En particulier, les informations suivantes, concernant les avions qui transitent par l'espace aérien, sont stockées dans cette vue.

Un avion est défini avant tout par son code transpondeur, son nom (Carthage, Sidi Bou Saï d, ...), sa capacité et sa date de mise en service. Chaque avion a une marque (Airbus A320, Boeing 767,...). Les routes suivie par les avions sont identifiées chacune par un numéro, un aéroport de départ et un aéroport d'arrivée. Deux routes de numéros différents peuvent avoir le même aéroport de départ ou le même aéroport d'arrivée.

Un aéroport est défini par un code, et la ville dans laquelle il se trouve, de même les villes sont identifiées par des codes en plus de leurs noms et leurs pays d'origine. Bien entendu, une ville donnée peut accueillir plusieurs aéroports et un pays donné regroupe plusieurs villes.

Un vol est défini par son indicatif (N° du vol), il est constitué d'un ensemble de tronçons correspondant aux escales effectuées lors de ce vol. Chaque tronçon est défini par la route suivie, la ville de départ, la ville d'arrivée, l'heure de départ et l'heure d'arrivée. L'ordre des tronçons est essentiel. Par exemple, le vol AF706 est constitué de 2 tronçons. Le premier tronçon relie Tunis et Paris suit la route 42 entre l'aéroport de Tunis-Carthage (situé à Tunis) et l'aéroport Roissy-Charles de Gaulle (situé à Paris). Le deuxième tronçon relie Paris à Athènes et suit la route 445 entre l'aéroport Roissy-Charles de Gaulle et l'aéroport Elefthérios Vénizelos (situé à Athènes). On distingue 3 sortes de vol :

- Les vols quotidiens
- Les vols hebdomadaires, ils ont lieu un jour particulier (Lundi, mardi,...,dimanche) de la semaine.
- Les vols charter ils se produisent à différentes dates spécifiques.

Les informations gérées par le SSGPV se limitent à ces aspects et seront stockées dans une base de données implantée sur un PC relié par le réseau local au SSGEA.

B/ QUESTIONS

Le travail demandé est décomposé en 4 parties qui peuvent être traitées de façon indépendante.

1^{ère} partie – Etude du Sous Système de Gestion de l'Espace Aérien

La spécification du SSGEA est conduite à l'aide de la méthode SA-RT selon l'approche Ward et Mellor. Une spécification partielle est fournie en annexe A2.

Dans les réponses aux questions, on respectera les identifiants précisés dans le dictionnaire de données.

Analyse

Question 1.1

- a- Partant de l'ébauche de spécification fournie en annexe A2, compléter le diagramme de la feuille de réponse N°1 qui montre la décomposition de la transformation de données 3.0 "Traiter Trame" à l'aide des transformations de données suivantes dont la description est fournie en annexe A2 :
- Associer Estampille
 - MAJ Paramètres de correction
 - Corriger Coordonnées
 - Examiner Code Transpondeur
 - Lire et maj param avion
 - Reconnaître et maj param avion
 - Transmettre paramètres
- b- Spécifier à l'aide d'un diagramme Etats-Transitions la transformation de contrôle 3.6 "Gérer Interprétation Trame".

Question 1.2

Fournir le schéma de transformation 2.0 qui décompose la transformation de données "vérifier alarmes"

Conception et Implémentation

Nous nous proposons d'implanter le système sur un PC muni de 3 cartes (voir figure 5) :

- une carte multi-canaux ,
- une carte modem pour la communication avec les SSLV à travers liaison une LS,
- une carte réseau pour l'accès à travers le réseau local au SSGPV.

Toutes ces cartes communiquent avec le microprocesseur à travers le BUS PCI de la carte mère. Chacune d'elle interrompt le microprocesseur à l'aide d'un signal d'interruption IRQ.

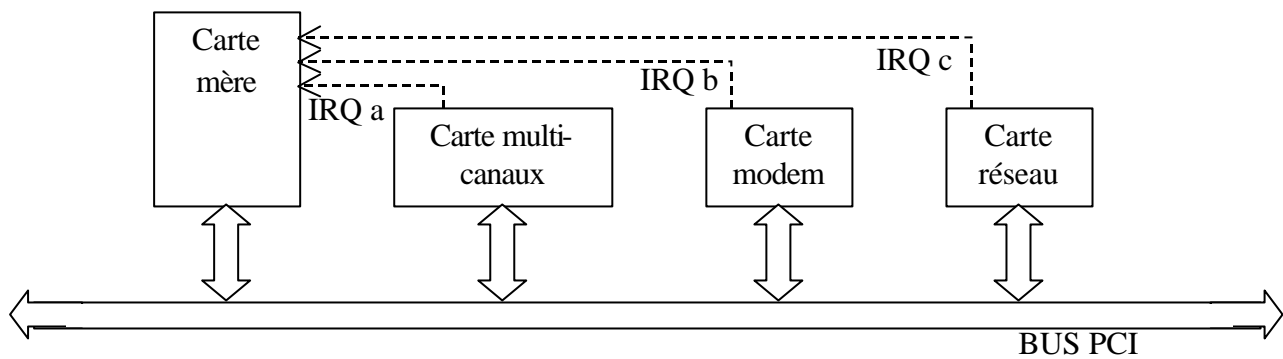


Figure 5 – architecture matérielle du SSGEA

Question 1.3

Effectuer les modifications nécessaires aux diagrammes fournis en annexe A2 (schéma de contexte et schéma préliminaire) pour tenir compte des choix matériels.

Question 1.4

Proposer une partition en tâches des différentes transformations de données et affecter à chaque tâche identifiée son niveau de priorité.

Question 1.5

Nous nous proposons d'implémenter la transformation de données 3.0 "Traiter Trame" à l'aide d'une seule tâche. Chacune des transformation de données du niveau inférieur sera implémentée par une fonction.

- a- Proposer un algorithme qui implémente cette transformation et qui montre l'appel aux différentes fonctions.
- b- Proposer un algorithme qui réalise le traitement associé à la transformation "Reconnaître trame".

2^{ème} Partie – Etude du Sous Système Local de Visualisation

L'étude du SSLV utilise le langage de modélisation UML. L'annexe A3 donne un récapitulatif des éléments de notation de ce formalisme. Dans toute la suite on se limitera aux classes correspondantes aux objets identifiés dans la partie 3.

Question 2.1

Donner le diagramme des classes pour le SSLV. Il conviendra d'explicitement uniquement les relations entre les classes. Il n'est pas demandé de donner les attributs ou les méthodes.

Question 2.2

Donner un diagramme de séquences qui montre le comportement du SSLV suite à l'arrivée d'un message émis par le SSGEA. On n'explicitera pas le traitement du message (cf. question 2.3)

Question 2.3

Donner un diagramme de collaboration qui montre le traitement du message PLANE_POSITION par le SSLV.

Question 2.4

Donner un diagramme de séquences qui montre le comportement du SSLV suite à une commande de modification de zoom faite par l'opérateur.

Question 2.5

Donner un diagramme de séquences qui montre le comportement du SSLV suite à une commande de changement de route d'un avion faite par l'opérateur.

Question 2.6

Déterminer les classes qui doivent disposer d'un flot de contrôle indépendant.

3^{ème} partie - Etude de la communication entre SSGEA et SSLV

On se propose d'implémenter partiellement les processus assurant la communication selon un schéma client-serveur. Pour cela, l'aspect communication de l'objet « Listener » sera assuré par un client C1 et celle de l'objet « Transmitter » par un client C2.

Question 3.1

Donner le squelette d'une application client/serveur sous Unix en mode connecté puis en mode non connecté.

Question 3.2

Quels sont les principales différences entre les deux modes de connexion au niveau du protocole TCP/IP (UDP et TCP) ? Lequel choisirez-vous pour cette application ? Justifiez ?

Question 3.3

Ecrire le code des processus serveur S1 et client C1 qui seront installés respectivement au niveau du sous système SSGEA et du sous système SSLV afin de permettre la communication dans le premier sens.

Question 3.4

Ecrire le code des processus serveur S2 et client C2 qui seront installés respectivement au niveau du sous système SSGEA et du sous système SSLV afin de permettre la communication dans le deuxième sens.

Question 3.5

Au niveau de chaque SSLV, la prise en compte de la commande changement de route (incluant l'activation du processus C2) a lieu suite à une interruption clavier. Donner la partie du code à ajouter à l'application installée au niveau du SSLV pour permettre ce mode de fonctionnement.

4^{ème} partie - Etude du Sous Système de Gestion des Plans de Vol

Question 4.1

Donner le modèle Conceptuel de données (MCD) normalisé (Modèle entités-associations) de la base.

Question 4.2

En déduire le modèle relationnel de la base.

Question 4.3

Donner les requêtes en SQL permettant de répondre aux questions suivantes :

- a- Quel est le N° du vol et le N° de la route suivie par un avion ayant le code transpondeur C donné.
- b- Quels sont les numéros des vols charters qui empruntent une route de numéro R donné à une date D donnée.
- c- Quel est le nombre de vols, par aéroport, atterrissants dans une ville de code V donné à une date D donnée.
- d- Quel est la liste des villes traversées par un vol de Numéro V donné. Par exemple, pour le vol AF 706 (cf. §5), la requête fournira le résultat suivant TUNIS-PARIS-PARIS-ATHENES.

C/ Annexes

Annexe A1 – **Système de coordonnées cylindriques**

Annexe A2 – **Spécification partielle par SA-RT**

Annexe A3 – **Éléments de langage de modélisation UML**

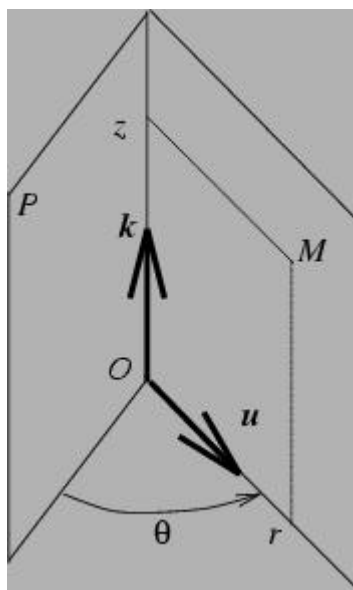
Annexe A4 – **Les sockets**

Annexe A1 - Coordonnées Cylindriques

Dans l'espace, un système de coordonnées cylindriques est formé par :

- un point O appelé *origine*,
- une direction donnée par un vecteur unitaire \mathbf{k}
- et un plan de référence angulaire P passant par O et contenant \mathbf{k} .

Soit M le point à repérer. On appelle *plan méridien* le plan passant par M , O et \mathbf{k} . Le plan méridien est repéré par son angle avec le plan de référence P , le point M est repéré dans le plan méridien par ses coordonnées cartésiennes (r,z) sur le repère orthonormé $\{O, \mathbf{u}, \mathbf{k}\}$.



Le point M est alors repéré par ses coordonnées (r, z) :

Annexe A2 - Spécification partielle par SA-RT

Cette annexe fournit la spécification partielle du SSGEA par la méthode SA-RT selon l'approche Ward et Mellor.

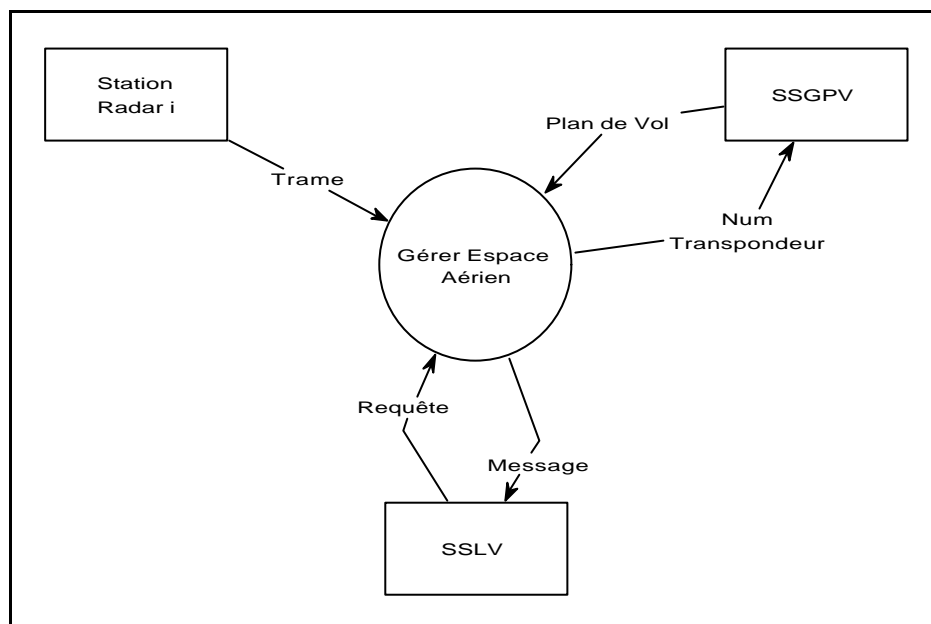


Schéma de contexte

Dictionnaire de données

This report contains an alphabetic list of all dictionary items and any details which have been filled in.

Name: Avion en Cours Type: Discrete flow	Name: Message Alarme Type: Discrete flow Bnf: ID_Avion + Type_Alarme
Name: avion en vue Type: Discrete flow	Name: Messages Type: Store Bnf: = PLANE_ENTER / PLANE_POSITION / PLANE_LEFT
Name: Coordonnées Type: Discrete flow	Name: Messages Type: Discrete flow
Name: Corrdonnées CPME Type: Store Comment: Coordonnées des différents CPME	Name: Non identifié Type: Control flow
Name: Corrigé Type: Control flow	Name: Nouveau Type: Discrete flow
Name: CPME Type: Control flow	Name: Num Transpondeur Type: Discrete flow
Name: Delta_h Type: Discrete flow	Name: Num_Vol Type: Discrete flow
Name: Delta_r Type: Discrete flow	Name: OK estampille Type: Control flow
Name: Delta_t Type: Discrete flow	Name: OK lecture Type: Control flow
Name: Direction Type: Discrete flow	Name: OK MAJ Type: Control flow
Name: fin reconnaître Type: Control flow	Name: Paramètre Avion en Cours Type: Store Bnf: ID_Avion + Nouveau + Reconnu + Num_Vol + Position + Vitesse + Direction @Position = Position_r + Position_t + Position_h @Direction = Direction_r + Direction_t + Direction_h @Nouveau = 1 0
Name: ID_Avion Type: Discrete flow	Name: Paramètre Avion en Cours Type: Discrete flow
Name: Liste Avions en Vue Type: Store Bnf: 0 {avion en vue} n @avion en vue = Code transpondeur + ID_Avion + N°vol +Dernière Position + (vitesse) + (Direction) + Route + date dernière detection Comment: contient aussi le dernier point détecté, vitesse estimée, direction	Name: Paramètres de correction Type: Store Bnf: Delta_r + Delta_t + Delta_h
Name: Liste Avions en Vue Type: Discrete flow	Name: Paramètres de correction Type: Discrete flow
Name: Message Type: Discrete flow	

Name: Plan de Vol Type: Discrete flow	Name: Requête Type: Discrete flow
Name: Plan vol Nouvel Avion Type: Discrete flow	Name: Trame Type: Discrete flow
Name: Plane_left Type: Discrete flow	Name: Trame Estampillée Type: Store
Name: Position Type: Discrete flow	Name: Type_Alarme Type: Discrete flow Bnf: E / H / R / D Comment: E = Ecart, H = Hauteur, D= Disparition, R=Rapprochement
Name: Reconnu Type: Control flow	

Description des transformations de données

Name: Associer Estampille
Comment: Associer une estampille temporelle correspondant à l'instant de réception de la trame et change les coordonnées par rapport à un repère commun

Name: Construire Message
Comment: Construit le message a envoyer au SSLV en fonction des paramètres qui se trouvent dans "Avion en cours". Elle peut générer les message PLANE_ENTER et PLANE_POSITION

Name: Corriger Coordonnées
Comment: Corrige les coordonnées de la trame reçue en fonction des paramètres de correction

Name: Emettre Message
Comment: Emet les différents messages en donnant la priorité aux messages d'alarmes

Name: Examiner Code Transpondeur

Name: Interroger SSGPV
Comment: s'occupe de l'interrogation du SSGPV pour obtenir les plan de vol des différents avions

Name: Lire et maj param avion
Comment: en fonction du code transpondeur de l'avion, cette transformation de données met à jour les différents champs du message à envoyer dans " Paramètres Avion en cours" ainsi que les données du même avion dans la "liste des avions en vue".

Si l'avion est nouveau et n'est pas dans la liste des avions en vue, cette transformation fournit le numéro transpondeur pour obtenir le plan de vol du nouvel avion et effectue son traitement en remplissant les champs correspondant dans la structure de données de "Paramètres Avion en cours". Elle lui associe également un ID_Avion

Name: MAJ Paramètres de correction
Comment: Mise à jour des paramètres de correction de coordonnées en fonction des coordonnées connues du CPME

Name: Modifier Route

Comment: suite à une requête de modification de route elle met à jour le stock de données

Name: Reconnaître et maj param avion

Comment: Est utilisée quand le code transpondeur de l'avion n'est pas connu. connaissant les dernières positions, les vitesses et les directions des différents avions en vue, cette transformation, recherche à quel avion appartient la trame en cours de traitement et met à jour les différents champs du message à envoyer dans " Paramètres Avion en cours" ainsi que les données du même avion dans la "liste des avions en vue".
Si c'est un nouvel avion, elle lui associe un ID_Avion.

Name: Transmettre Paramètres

Comment: Quand tous les paramètres du message concernant l'avion en cours sont recueillis cette transformation de données permet de transmettre le message a envoyer au SSLV

Name: Vérifier Alarmes

Comment: vérifie les différentes situation d'alarmes et alerte le cas échéant la transformation de contrôle. Elle décide de transmettre un message PLANE_LEFT ou un message d'alarme Disparition en fonction de la position de l'avion

Annexe A3 – Eléments de langage de modélisation UML

Annexe A4 – Les sockets

1 Communication par le réseau TCP-IP

1.1 Sockets, addresses

```
#include <sys/types.h>
#include <sys/socket.h>
```

```
int socket(int domain, int type, int protocol);
```

Pour communiquer, les applications doivent créer des *sockets* (prises bidirectionnelles) par la fonction `socket()` et les relier entre elles. On peut ensuite utiliser ces sockets comme des fichiers ordinaires (par `read`, `write`, ...) ou par des opérations spécifiques (`send`, `sendto`, `recv`, `recvfrom`, ...).

Pour désigner un socket sur une machine il faut une *adresse de socket*. Comme il existe différents types de sockets, les opérations sur les addresses concerne un type d'adresse général abstrait (`struct sockaddr`) qui recouvre tous les types concrets particuliers.

Pour TCP-IP (Inet), les addresses de sockets sont déterminées par un numéro IP, et un numéro de port. Pour IPv4, on utilise des `struct sockaddr_in`, qui possèdent 3 champs importants :

- `sin_family`, la famille d'addresses, valant `AF_INET`
- `sin_addr`, pour l'adresse IP.
- `sin_port`, pour le numéro de port

Attention, les octets de l'adresse IP et le numéro de port sont stockés dans *l'ordre réseau* (big-endian), qui n'est pas forcément celui de la machine hôte sur laquelle s'exécute le programme. Voir plus loin les fonctions de conversion hôte/réseau.

Pour IPv6, on utilise des `struct sockaddr_in6`, avec

- `sin6_family`, valant `AF_INET6`
- `sin6_addr`, l'adresse IP sur 6 octets
- `sin6_port`, pour le numéro de port.

1.2 Remplissage d'une adresse

Quand on ouvre un socket local, il faut le créer par `socket()` (voir plus loin) et «nommer la prise», c'est-à-dire lui associer (par `bind()`) une adresse. De même, on a aussi besoin d'addresses pour désigner les sockets distants auxquels on veut se connecter.

Préparation d'une adresse distante

Dans le cas le plus fréquent, on dispose du nom de la machine destinataire (par exemple la chaîne de caractères "www.mes.tn"), et du numéro de port (un entier).

```
#include <netdb.h>
struct hostent *gethostbyname(const char *name);
```

`gethostbyname()` retourne un pointeur vers une structure `hostent` qui contient diverses informations sur la machine en question, en particulier une adresse `h_addr`

Une machine peut avoir plusieurs addresses que l'on mettra dans `sin_addr`.

Le champ `sin_port` est un entier court *en ordre réseau*, pour y mettre un entier ordinaire il faut le convertir par `htons()`
Host TO Network Short

Préparation d'une adresse locale

Pour une adresse locale, on utilise

- dans le cas le plus fréquent, l'adresse `INADDR_ANY`. Le socket est alors ouvert (avec le même numéro de port) sur toutes les adresses IP de toutes les interfaces de la machine.
- l'adresse `INADDR_LOOPBACK` correspondant à l'adresse locale `127.0.0.1`. Le socket n'est alors accessible que depuis la machine elle-même.
- une des adresses IP de la machine.

Examen d'une adresse

La fonction `getsockname()` permet de retrouver l'adresse associée à un socket.

```
#include <sys/socket.h>
int getsockname(int s ,
                struct sockaddr *name ,
                socklen_t * namelen )
```

Le numéro de port est dans le champ `sin_port`, pour le convertir en entier normal, utiliser `ntohs()` (Network TO Host short).

L'adresse IP peut être convertie en chaîne de caractères en notation décimale pointée (exemple "193.95.69.11") par `inet_ntoa()` (network to ascii),

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

char *inet_ntoa(struct in_addr in);
```

On peut également tenter une *résolution inverse*, c'est-à-dire de retrouver le nom à partir de l'adresse, en passant par `gethostbyaddr`, qui retourne un pointeur vers une structure `hostent`, dont le champ `h_name` désigne le nom officiel de la machine.

Cette résolution n'aboutit pas toujours, parce que tous les numéros IP ne correspondent pas à des machines déclarées.

```
#include <netdb.h>
extern int h_errno;

struct hostent *gethostbyaddr(const char *addr, int len, int
type);

struct hostent {
    char *h_name; /* official name of host */
    char **h_aliases; /* alias list */
    int h_addrtype; /* host address type */
    int h_length; /* length of address */
    char **h_addr_list; /* list of addresses */
}
#define h_addr h_addr_list[0] /* for backward compatibility
*/
```

1.3 Fermeture d'un socket

Un socket peut être fermé par `close()` ou par `shutdown()`.

```
int shutdown(int fd, int how);
```

Un socket est bidirectionnel, le paramètre `how` indique quelle(s) moitié(s) on ferme : 0 pour la sortie, 1 pour l'entrée, 2 pour les deux (équivalent à `close()`).

1.4 Communication par datagrammes (UDP)

Création d'un socket

```
#include <sys/types.h>
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

Cette fonction construit un socket et retourne un numéro de descripteur. Pour une liaison par datagrammes sur IPv4, utilisez la famille `AF_INET`, le type `SOCK_DGRAM` et le protocole par défaut 0.

Retourne -1 en cas d'échec.

Connexion de sockets

La fonction `connect` met en relation un socket (local) avec un autre socket désigné, qui sera le «correspondant par défaut» pour la suite des opérations.

```
#include <sys/types.h>
#include <sys/socket.h>

int connect(int sockfd,
            const struct sockaddr *serv_addr,
            socklen_t addrlen);
```

Envoi de datagrammes

Sur un socket connecté (voir ci-dessus), on peut expédier des datagrammes (contenus dans un tampon `t` de longueur `n`) par `write(sockfd, t, n)`.

La fonction `send()`

```
int send(int s, const void *msg, size_t len, int flags);
```

permet d'indiquer des *flags*, par exemple `MSG_DONTWAIT` pour une écriture non bloquante.

Enfin, `sendto()` envoie un datagramme à une adresse spécifiée, sur un socket connecté ou non.

```
int sendto(int s, const void *msg, size_t len, int flags,
           const struct sockaddr *to, socklen_t tolen);
```

Réception de datagrammes

Inversement, la réception peut se faire par un simple `read()`, par un `recv()` (avec des *flags*), ou par un `recvfrom`, qui permet de récupérer l'adresse `from` de l'émetteur.

```
int recv(int s, void *buf, size_t len, int flags);
int recvfrom(int s, void *buf, size_t len, int flags,
             struct sockaddr *from, socklen_t *fromlen);
```

1.5 Communication par flots de données (TCP)

La création d'un socket pour TCP se fait ainsi

```
int fd;  
..  
fd=socket(AF_INET, SOCK_STREAM, 0);
```

Programmation des clients TCP

Le socket d'un client TCP doit être relié (par `connect()`) à celui du serveur, et il est utilisé ensuite par des `read()` et des `write()`, ou des entrées-sorties de haut niveau `fprintf()`, `fscanf()`, etc. si on a défini des flots par `fdopen()`.

Réaliser un serveur TCP

Un serveur TCP doit traiter des connexions venant de plusieurs clients.

Après avoir créé et nommé le socket, le serveur spécifie qu'il accepte les communications entrantes par `listen()`, et se met effectivement en attente d'une connexion de client par `accept()`.

```
#include <sys/types.h>  
#include <sys/socket.h>  
  
int listen(int s, int backlog);  
int accept(int s, struct sockaddr *addr,  
           socklen_t *addrlen);
```

Le paramètre `backlog` indique la taille maximale de la file des connexions en attente. Sous Linux la limite est donnée par la constante `SOMAXCONN` (qui vaut 128), sur d'autres systèmes elle est limitée à 5.

La fonction `accept()` renvoie un autre socket, qui servira à la communication avec le client.

L'adresse du client peut être obtenue par les paramètres `addr` et `addrlen`.

En général les serveurs TCP doivent traiter plusieurs connexions simultanément. La solution habituelle est de lancer, après l'appel à `accept()` un processus fils (par `fork()`) qui traite la communication avec un seul client. Ceci induit une gestion des processus, donc des signaux liés à la terminaison des processus fils.

SIGNATURES DES
SURVEILLANTS

DISCIPLINE :OPTION :

EPREUVE DE :

SIGNATURE DU CANDIDAT

NOM : PRENOMS :

N° :N°/C.I.N.

N° PAGE/.....

EPREUVE DE:

Document Réponse N°1

Associer
Estampille

8

Examiner Code
Transpondeur

3

MAJ Paramètres
de correction

5

Gérer Interpr-
étation Trames

6

Lire et maj
param avion

4

Corriger
Coordonnées

2

Reconnaître
et maj param
avion

1

Transmettre
Paramètres

7