

Simulation d'un système de sms avec accusé de réception

Mini projet IN301/IN3ST01 - 2007-2008

Sujet rédigé par Laurent Najman,

très librement inspiré d'un projet de Jérôme Gueydan pour l'ENSTA

26 novembre 2007

Table des matières

1	Introduction	4
1.1	Le principe des sms	4
1.2	Le routage	5
2	Cahier des charges	6
2.1	Cahier des charges fonctionnelles	6
2.1.1	Architecture fonctionnelle	6
2.1.2	Le téléphone	7
2.1.3	Serveur central	7
2.1.4	Le réseau intergalactique	7
2.2	Cahier des charges techniques	8
2.2.1	Contraintes générales	8
2.2.2	Mémoire des serveurs centraux	8
2.2.3	Nombre de processus	8
2.2.4	Paramètres	8
2.2.5	Gestion des échanges par tuyaux	9
2.3	Comment tester sans s'y perdre ?	9
2.4	Livrables	9
3	Conduite du projet	10
3.1	Introduction	10
3.1.1	Un projet en plusieurs étapes	10
3.1.2	Méthode "diviser pour régner"	10
3.2	Etape 1 : les fonctions de communication	11
3.3	Etape 2 : réalisation de programmes indépendants	11

3.3.1	Processus : <i>Central</i>	11
3.3.2	Processus : <i>Telephone</i>	11
3.4	Etape 3 : création d'un réseau central	12
3.4.1	Préparation de la communication par tuyaux	12
3.4.2	Raccordement	12
3.5	Etape 4 : création du réseau intergalactique	12
3.5.1	Processus : <i>Intergalactique</i>	12
3.5.2	Raccordement	13
4	Évolutions complémentaires et optionnelles	13
4.1	Identification des sms	13
4.2	Des numéros de téléphones inconnus	13
4.3	Des annuaires partiels	13
4.4	Utilisation de socket	14
5	Annexes	14
5.1	De l'intérêt des standards pour assurer une meilleure interopérabilité	14
5.2	Spécifications de protocoles de communication	14
5.2.1	Un format de message unique	14
5.2.2	Protocole sur le réseau	15
5.3	Générateur aléatoire et manipulation de chaînes en C	16
5.4	Redirection des entrées et des sorties	16

Avertissement

Le projet suivant est à faire par groupe de deux étudiants (en une séance de TD de 2h, deux séances de TP de 3h, 15h de projet encadré, plus un **petit** travail personnel si nécessaire), et à rendre pour janvier 2008, à une date qui sera précisée ultérieurement.

Pour faire court, sachez que tous les projets "semblables" auront une note de zéro¹, et que les auteurs seront convoqués en conseil de discipline. Ceux qui arriveront à faire la preuve qu'ils ont écrit le projet duquel les autres se sont inspirés n'auront aucune pénalité supplémentaire par rapport à la note de zéro. Si vous faites le projet à plus de deux, un coefficient de réduction proportionnel sera appliqué. Par exemple, si vous faites le projet à trois, la note sera réduite d'un tiers. Aucun bonus ne sera donné pour un étudiant travaillant seul.

Plagiat

Le plagiat est l'utilisation, sans citation appropriée, du travail intellectuel d'une autre personne dans un travail soumis à évaluation. Ce qui suit est très fortement inspiré et traduit de la page

<http://www.csd.abdn.ac.uk/teaching/handbook/both/info.php?filename=cheating.txt>.

¹Il existe des logiciels de détection automatique de plagiat.

Quand vous écrivez un rapport ou du code qui contient des parties ou qui paraphrase le travail d'autres personnes, vous devez clairement le signaler. En particulier, les citations doivent être données entre guillemets, avec les références appropriées.

1. Le code soumis pour évaluation doit clairement être annoté avec le nom de l'étudiant qui a soumis l'exercice, avec la date de rédaction.
2. Quand un exercice contient du code qui n'a pas été écrit par l'étudiant qui soumet le travail, ou contient du code écrit par l'étudiant lui-même à un autre moment, le code en question doit être clairement identifié et annoté avec le nom de l'auteur, le copyright (si différent), la date d'achèvement ou de publication, et une référence à la source (par exemple une URL ou une référence bibliographique).
3. En principe, la discussion avec d'autres étudiants du contenu du cours et des exercices non-évalués est encouragée, car une telle discussion amène généralement à une meilleure compréhension du sujet. Cependant, les discussions doivent rester à un niveau général, et ne doivent pas descendre à un niveau détaillé de conception ou de codage. Le travail soumis **pour évaluation doit être un travail individuel**. Si vous soumettez un travail produit conjointement avec une autre personne, ou qui inclut le travail de quelqu'un d'autre, **ceci doit être clairement indiqué, et le code en question doit être clairement identifié**. L'évaluation du travail se fera sur la base de la valeur ajoutée par l'étudiant.
4. De manière identique, bien que nous encourageons la ré-utilisation de logiciel existant comme une bonne pratique d'Ingénierie Logicielle, les origines d'une telle ré-utilisation doivent **être clairement indiquées, et le code identifié**. L'évaluation du travail se fera sur la base de la valeur ajoutée par l'étudiant.
5. Quand le travail est à réaliser en groupe de deux étudiants, le rapport écrit ou le travail doit clairement identifier et distinguer ce qui a été réalisé par un des étudiants, ou ce qui a été fait par le groupe en entier.

Consignes

- Vous devez m'envoyer avant janvier 2008 dernier délai (la date effective sera précisée ultérieurement) une archive comprimée en tar.gz, contenant le code, le makefile, et les rapports du projet sous format électronique. Les rapports papiers doivent être déposés le même jour au secrétariat A2SI. Un barème de "moins deux points" par jour de retard sera appliqué. **Une archive qui ne sera ni du tar.gz ni du .zip ne sera pas corrigée.**
- Le projet devra se compiler en exécutant la commande make sur une machine sous Linux. **Un projet qui ne respectera pas cette consigne ne pourra pas être corrigé.**
- Le rapport du projet décrira les diverses solutions aux différents problèmes rencontrés, et **justifiera** la solution choisie.
- Votre note dépendra de la qualité de votre rapport, et de la qualité de votre code.
- Le barème prend en compte pour une part la rédaction de la réponse, et également l'écriture du code correspondant aux choix que vous aurez faits. Vous pourrez donc avoir des points si vous n'avez pas programmé vos propositions, mais vous

n'aurez pas forcément tous les points si le code que vous avez écrit n'est pas proprement justifié.

Les consignes de remise du projet seront précisées sur la page web du cours. D'autres éléments, comme le barème d'évaluation, y seront également notés.

Remarques

L'énoncé peut sembler épais de prime abord. Ce n'est pas le cas : il se résume aux pages 4 à 9. Le reste du document a pour objectif de vous aider dans la réalisation du projet, en indiquant des éléments de solutions, en attirant l'attention sur les points essentiels, et en donnant une démarche saine de progression.

Cet exercice est avant tout "académique", mais n'est pas dénué d'intérêt ; en effet, des sociétés commercialisent toutes sortes de simulateurs pour tester le fonctionnement des nouveaux composants , afin de valider leur fonctionnement avant leur mise en production.

La rédaction du texte du sujet est inspiré par un projet anciennement proposé par Jérôme Gueydan pour l'ENSTA.

1 Introduction



Les compagnies de téléphones mobiles Béton, Bleue, et ElleSaitPasFaire, ont monté un consortium afin de mettre au point un nouveau système de sms fondé sur les toutes dernières technologies. En particulier, elle veulent que chaque message envoyé retourne un accusé de réception. Si au bout d'un certain temps, l'accusé de réception n'a pas été reçu, le message doit être réémis.

L'objectif du projet est de simuler ces échanges de sms entre téléphones et serveurs centraux. Pour cela, après mûre réflexion, elles ont décidé de faire appel à vos services. Vous recevez le cahier des charges ci-dessous.

Avant de présenter le sujet, examinons le fonctionnement des sms.

1.1 Le principe des sms

Le principe des messages par sms met en relation plusieurs acteurs :

- le client émetteur (et son téléphone), qui souhaite envoyer un sms ;
- le central de la compagnie de téléphone du client (par exemple, Béton) à laquelle est connectée le téléphone ;
- le central de la compagnie de téléphone du récepteur du sms (par exemple, Bleue) ;
- le client récepteur (et son téléphone), qui doit recevoir le sms ;

Les téléphones sont reliés à leur compagnie par la toute dernière technologie 2Gmax. Les compagnies sont reliées entre elles par un réseau dédié, le *réseau intergalactique* (voir fig. 1).

Supposons maintenant que le client lambda souhaite émettre un sms. Les opérations suivantes ont lieu :

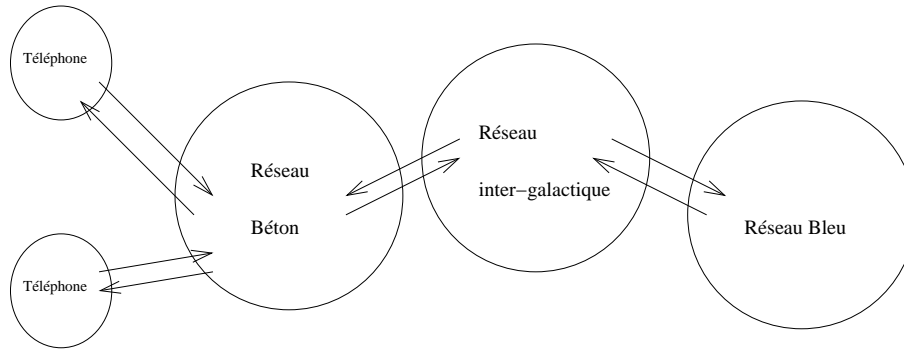


FIG. 1 – Principe de communication des téléphones et des différents réseaux

1. Le téléphone se connecte sur le central de son réseau Béton, et émet le sms.
2. Le central de Béton regarde le numéro de téléphone vers lequel le sms doit être acheminé. Si ce numéro était dans son réseau, il transmettrait le sms au bon téléphone. Mais ce numéro n'est pas dans son réseau, il achemine donc le sms vers un autre réseau, le réseau bleu, au travers du réseau intergalactique. Comme il veut rémettre le message si il ne reçoit pas l'accusé de réception, il doit garder trace du sms.
3. Le réseau bleu prend connaissance du sms, et l'achemine vers le destinataire. Il attend le retour du téléphone récepteur, comme quoi tout s'est bien passé. Ensuite, il achemine l'accusé de réception vers le central de Béton, toujours par le réseau intergalactique.
4. Enfin, le central de Béton achemine l'accusé de réception vers le téléphone émetteur. Il peut alors effacer la trace du sms original qu'il conservait.
5. Si le central de Béton n'a pas reçu d'accusé de réception au bout d'un certain temps, il décide de rémettre le sms. Si au bout de trois émissions, il n'a reçu aucun accusé de réception, le sms est déclaré perdu, et le central envoie cette information au téléphone émetteur.

1.2 Le routage

Pour effectuer le routage des sms, c'est-à-dire pour déterminer à quel serveur central chaque sms doit être transmis, le serveur utilise un annuaire des numéros de téléphone.

Chaque serveur analyse donc le numéro de téléphone du destinataire, puis :

- si l'émetteur est dans le même réseau que le destinataire (et que le serveur), il envoie le sms directement au téléphone concerné ;
- si le destinataire est dans une autre banque, le serveur envoie la demande sur le réseau intergalactique, sans se préoccuper de la suite du transit.

Le réseau intergalactique n'est donc pas un simple réseau physique : il doit aussi effectuer le routage des sms, c'est-à-dire analyser les messages qui lui sont transmis,

envoyer chaque demande vers le serveur du réseau correspondant et, enfin, prendre en charge la transmission de la réponse lorsqu'elle lui revient.

2 Cahier des charges

L'objectif de ce projet est de simuler les mécanismes décrits ci-dessus, c'est-à-dire :

- le téléphone envoyant un sms au serveur central ;
- le serveur effectuant le routage du sms vers lui-même ou le réseau intergalactique, et effectuant le routage des accusés de réception qu'il reçoit ;
- le serveur gardant trace des sms qui sont émis par ses téléphones, et qui émet à nouveau les sms dont il n'a pas reçu d'accusé de réception ;
- le réseau intergalactique auquel sont connectés les différents serveurs centraux, capable d'effectuer le routage des sms et des accusés de réception ;
- le tout permettant un maximum de parallélisme.

Le cahier des charges fonctionnelles précise l'architecture générale, les fonctionnalités devant être programmées ainsi que les contraintes fonctionnelles à respecter. Le cahier des charges techniques fournit les restrictions concernant la mise en œuvre.

2.1 Cahier des charges fonctionnelles

2.1.1 Architecture fonctionnelle

Le schéma 2 précise celui qui a été présenté plus haut et retranscrit la description que nous avons fournie ci-dessus.

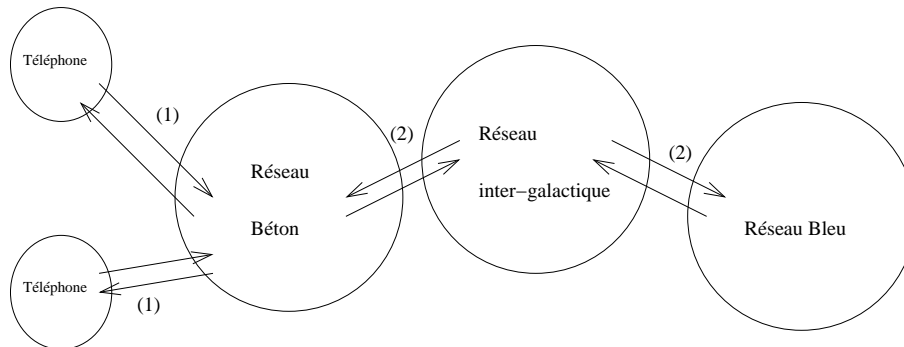


FIG. 2 – Architecture fonctionnelle du projet

Chaque téléphone est relié par 25Gmax (1) au serveur central du réseau de l'émetteur.

Le réseau intergalactique relie (2) les serveurs centraux des différentes compagnies. Tous les autres serveurs centraux des compagnies de la place sont également reliées au réseau intergalactique, mais ne sont pas représentées sur ce schéma.

2.1.2 Le téléphone

Dans le cadre de ce projet, il n'est pas question d'utiliser de vrais téléphones. Un téléphone étant un moyen d'envoyer un sms, il sera simulé pour ce projet par un exécutable qui enverra un message aléatoire d'un certain nombre de caractères.

Pour fonctionner, un téléphone a donc besoin de connaître la liste des numéros de téléphones existants².

Chaque téléphone devra envoyer les sms générés vers son serveur.

Si un téléphone reçoit un sms, il l'affiche à l'écran, avec le numéro de l'émetteur. Dans le cours normal des événements, le téléphone doit alors retourner un accusé de réception. Pour les besoins du projet, nous simulerons le fait qu'un téléphone peut ne pas recevoir de sms de la manière suivante : un téléphone ne retournera un accusé de réception qu'avec une certaine probabilité. Cette probabilité sera donnée en paramètre au lancement du programme.

Lorsqu'un téléphone reçoit un accusé de réception, il affiche ce message à l'écran (*sms reçu ou perdu*).

Les échanges entre le téléphone et leur serveur ont lieu suivant un protocole bien déterminé : les informations sont formatées d'une certaine façon. Afin de simplifier le projet et de garantir l'interopérabilité des différents projets entre eux (voir en annexe pour une explication de ce point), un seul protocole de communication sera utilisé. Celui-ci est décrit en annexe de ce document.

2.1.3 Serveur central

Un serveur central a une fonction de routage et une fonction de renvoi périodique :

- il doit pouvoir accepter des messages provenant de téléphones et du réseau intergalactique ;
- il doit pouvoir effectuer le routage des sms et des accusés de réception vers le téléphone concerné ou bien vers le réseau intergalactique ;
- il doit pouvoir accepter les accusés de réception du réseau intergalactique ou des téléphones ;
- il doit pouvoir rémettre (trois fois au maximum) les sms dont il n'aurait pas reçu d'accusé de réception.

Un serveur central doit être capable d'utiliser le protocole de communication employé par les téléphones, ainsi que le protocole du réseau intergalactique (voir les protocoles définis en annexe).

Afin de pouvoir effectuer correctement le routage des messages, le serveur d'acquisition doit connaître l'annuaire des téléphones.

Un serveur central doit garder trace des messages qui passent par lui, et maintenir cette trace à jour.

2.1.4 Le réseau intergalactique

Le réseau intergalactique n'a qu'une fonction de routage :

²A proprement parler, ce n'est pas indispensable : il est parfaitement envisageable que le téléphone envoie un sms vers un numéro inexistant. Nous ne considérerons pas cette possibilité dans cette première approche.

- il doit pouvoir accepter les messages provenant des serveurs centraux ;
- il doit pouvoir analyser le contenu des messages pour déterminer vers quel serveur central il doit les envoyer.

Pour fonctionner, le réseau intergalactique doit posséder l’annuaire des téléphones.

2.2 Cahier des charges techniques

2.2.1 Contraintes générales

- L’ensemble de la simulation tournera sur une seule machine.
- Les programmes seront écrits en langage C, et mettront en œuvre les concepts et les techniques apprises pendant le cours IN301/IN3ST01.
- Un maximum de parallélisme est exigé dans ce projet.

2.2.2 Mémoire des serveurs centraux

Pour gérer les sms en attente, les serveurs centraux ont une mémoire finie. La taille de cette mémoire (le nombre de cases du tableau) sera précisée sur la ligne de commande au lancement d’un serveur.

2.2.3 Nombre de processus

Chaque composant “fonctionnel” tel qu’il a été présenté dans le cahier des charges fonctionnelles correspondra à un processus. On trouvera donc un processus pour le téléphone (que l’on nommera *Telephone*), un processus pour un serveur central (*Central*), et un processus pour le réseau intergalactique (*Intergalactique*).

La simulation pouvant mettre en jeu plusieurs téléphones et plusieurs centraux, etc., on trouvera en fait un processus *Telephone* par terminal et un processus *Central* par serveur central.

Pour favoriser le parallélisme, chaque processus pourra (si besoin est) être découpé en plusieurs threads.

2.2.4 Paramètres

Les paramètres nécessaires au fonctionnement des différents processus seront fournis via “la ligne de commande”, à l’exception des paramètres suivants qui seront fournis sous forme de fichier :

- Un annuaire des serveurs : la liste des serveurs centraux et de leur code à 2 chiffres (par exemple 01 pour bleu, 02 pour Béton, etc.) ;
- Un annuaire global des téléphones : la liste des téléphones de chaque serveur central et le serveur associé codé sur 2 chiffres.

Ces fichiers seront au format texte, chaque ligne contenant les deux informations demandées (nom du serveur et code de 2 chiffres, ou numéro de téléphone et serveur associé codé sur 2 chiffres), séparées par un espace.

2.2.5 Gestion des échanges par tuyaux

Les téléphones connectés au même serveur central utiliseront chacun une paire de tuyaux (qui seront des *tubes anonymes*) pour dialoguer avec ce serveur. Le serveur aura donc comme rôle d'orchestrer simultanément les lectures et les écritures sur ces tuyaux. Ceci implique de maintenir une table de routage entre téléphones pour chacun des processus *Central*.

En ce qui concerne le réseau intergalactique et les serveurs centraux, la problématique est strictement identique à celle des téléphones : il faudra utiliser une paire de tuyaux pour connecter chaque serveur central au réseau intergalactique. Ceci implique de maintenir une table de routage entre les serveurs pour le processus *Intergalactique*.

2.3 Comment tester sans s'y perdre ?

Le schéma proposé ci-dessus comporte un petit défaut tant que les communications entre processus se font sur la même machine (donc sans utiliser de *socket*, développement proposé en option) : chaque *Telephone* va fonctionner à partir de la même fenêtre *xterm* (le même terminal, le même *shell*). Tous les affichages vont donc se faire sur cette même fenêtre.

Afin de faire bénéficier chaque processus *Telephone* d'une entrée et d'une sortie standard qui lui soient propres, une astuce peut être utilisée : lors de la création d'un nouveau *Telephone* par le processus *Central*, recouvrir le nouveau fils du processus non pas par *Telephone*, mais par *xterm -e Telephone*.

Cette astuce n'est pas très esthétique, et n'est qu'un intermédiaire avant la communication par *socket*.

2.4 Livrables

Doivent être livrés sous format électronique :

1. Un ensemble de fichier ".c", amplement commentés, correspondant aux différents programmes constituant la simulation.
2. Un fichier Makefile permettant de compiler tous les programmes (y compris les programmes de test).
3. un mode d'emploi expliquant comment obtenir une application opérationnelle, comment l'exécuter, et comment la tester.
4. Un manuel technique détaillant l'architecture, le rôle de chaque composant, expliquant comment tester le bon fonctionnement de façon indépendante et justifiant les choix techniques effectués. Ce manuel devra en particulier bien préciser comment le projet répond au cahier des charges (ce qu'il sait faire, ce qu'il ne sait pas faire, et mettre en exergue ses qualités et ses défauts).
5. Dans le manuel technique, on démontrera qu'il ne peut pas y avoir d'interblocage entre les différents processus, pas plus qu'entre les threads d'un même processus.

Tous les documents à produire s'adressent à des ingénieurs généralistes et les rédacteurs doivent donc veiller à donner des explications concises et claires.

Aucun fichier exécutable, fichier objet, fichier de sauvegarde, fichier superflu ou inutile ne devra être transmis avec les livrables.

Une version papier des rapports sera également remise au secrétariat.

3 Conduite du projet

Les étapes qui vous sont suggérées dans cette partie permettent une approche “serene” de la programmation de ce projet.

3.1 Introduction

3.1.1 Un projet en plusieurs étapes

Le développement du simulateur se déroule en plusieurs étapes. Ce découpage conduit le développeur à écrire des programmes indépendants les uns des autres, qui communiqueront entre eux par l’intermédiaire de tuyaux, souvent après redirection des entrées/sorties standards.

3.1.2 Méthode “diviser pour régner”

La constitution de l’application globale par écriture de “petits” programmes indépendants qui communiqueront ensuite entre eux est un gage de réussite : chaque “petit” programme générique peut être écrit, testé et validé indépendamment, et la réalisation du projet devient alors simple et progressive.

La meilleure stratégie à adopter consiste à écrire une version minimale de chaque brique du projet, à faire communiquer ces briques entre elles, puis éventuellement, à enrichir au fur et à mesure chacune des briques. La stratégie consistant à développer à outrance une des briques pour ensuite s’attaquer tardivement au reste du projet conduit généralement au pire des rapport résultat/travail.

Avant toute programmation, conception ou réalisation de ce projet, il est fortement conseillé de lire l’intégralité de l’énoncé, y compris les parties que vous pensez ne pas réaliser, et de s’astreindre à comprendre la structuration en étapes proposée ici.

La traduction de cet énoncé sous forme de schéma est indispensable.

En particulier, il est très important de définir dès le départ l’ensemble des flux de données qui vont transiter d’un processus à un autre, de déterminer comment il est possible de s’assurer que ces flux sont bien envoyés et reçus, puis de programmer les fonctions d’émission et de réception correspondantes. Ces fonctions seront ensuite utilisées dans tous les processus du projet.

Un schéma clair et complet associé à des communications correctement gérées garantissent la réussite de ce projet et simplifient grandement son développement.

3.2 Etape 1 : les fonctions de communication

La première étape a été faite pour vous. Elle consiste à programmer les différentes fonctions de communication permettant de lire et écrire les différents messages selon les protocoles définis en annexe.

Les problèmes de synchronisation seront abordés dans les étapes suivantes et il s'agit pour l'instant uniquement de traduire sous forme de programmes (de fonctions en C) les protocoles de communication : formatage de messages selon la structure définie en annexe, récupération des informations stockées sous cette forme, etc.

Ces fonctions sont téléchargeables en suivant le lien donné en annexe. Une fois que vous aurez lu le code, et compris son utilisation, vous serez à même de les utiliser ; ces fonctions sont utilisées par tous les processus constituant le projet.

3.3 Etape 2 : réalisation de programmes indépendants

Dans cette seconde étape, il s'agit de mettre en place les différentes briques du projet et de pouvoir les tester indépendamment.

3.3.1 Processus : *Central*

Le processus *Central* recevra des messages en provenance soit des téléphones, soit du réseau intergalactique. Ces messages seront redirigés ("routés") vers le réseau intergalactique ou bien vers les téléphones, conformément au cahier des charges. A cette phase du projet, on pourra utiliser :

- l'entrée ou la sortie standard pour simuler le dialogue avec le serveur central ("à la main") ;
- des fichiers dans lesquels le processus *Central* ira écrire lorsqu'il est supposé envoyer un message ;
- des fichiers dans lesquels le processus *Central* ira lire les messages lorsqu'il s'attend à recevoir ; ces derniers seront préparés "à la main".

Ces fichiers permettront de simuler les échanges avec les processus *Central* et *Intergalactique*.

Le programme devra accepter sur sa ligne de commande au moins un paramètre représentant la taille de la mémoire servant à la gestion des sms.

Dans une première phase, le processus *Central* traitera *séquentiellement* chaque téléphone ainsi que les messages en provenance ou à destination du réseau intergalactique. Dans une deuxième phase, on *paralélisera* au maximum les requêtes. La première phase doit être pensée et conçue sur le papier afin de pouvoir facilement passer à la deuxième phase. La première phase peut suffire pour passer à la suite du projet, mais elle est obligatoire et devra être rendue. On conservera toutes les versions des différentes phases du programme à des fins de test.

3.3.2 Processus : *Telephone*

Le processus *Telephone* offrira l'interface permettant d'envoyer et de recevoir les sms. Un téléphone lit son entrée standard et écrit sur sa sortie standard.

Afin de pouvoir tester le fonctionnement du *Telephone*, les sms à destination du serveur central pourront être écrits dans un fichier. Les réponses seront également lues dans un fichier qui aura été préparé à l'avance. On peut rediriger les entrées/sorties sur la ligne de commande de la manière suivante : *Telephone < entree.txt > sortie.txt*.

3.4 Etape 3 : création d'un réseau central

3.4.1 Préparation de la communication par tuyaux

En pratique, le processus *Central* et chaque processus *Telephone* vont communiquer par une paire unique de tuyaux. Une fois ces deux tuyaux créés, il est possible de modifier simplement le programme *Central* pour qu'ils utilisent non pas des fichiers mais ces deux tuyaux. Le processus *Telephone* n'a pas à être modifié.

3.4.2 Raccordement

Pour terminer la mise en place des communications au sein d'une même banque, il faut maintenant créer d'une part une paire de tuyaux entre *Central* et chaque *Telephone* (il y aura autant de paires de tuyaux que de téléphones), après avoir redirigé leurs entrées et sorties standards (voir annexe).

Modifier le programme *Central* pour qu'il accepte sur sa ligne de commande les paramètres suivants :

- la taille de la mémoire de gestion des sms.
- le nom du fichier *annuaire des serveurs* ;
- les 2 chiffres associés à ce central ;
- le nom d'un fichier *annuaire global des téléphones* ;

Créer les tuyaux nécessaires, opérer les clonages et recouvrements nécessaires pour créer les processus *Telephone* en nombre suffisant, sans oublier les redirections nécessaires des tuyaux avec les entrées et sorties standards.

3.5 Etape 4 : création du réseau intergalactique

3.5.1 Processus : *Intergalactique*

Chaque serveur central sera relié au processus *Intergalactique* par une paire de tuyaux. Celle-ci permettra à *Intergalactique* de recevoir les messages de demande d'autorisation et de transmettre les réponses en retour, après les avoir routés.

L'architecture à mettre en place entre *Intergalactique* et les différents processus *Central* sera similaire à celle mise en place entre chaque processus *Central* et les processus *Telephone* qui y sont reliés.

Dans un premier temps, les messages transitant par *Intergalactique* seront simplement lus sur l'entrée et la sortie standard (ou lus et écrits dans des fichiers, sans qu'aucune communication ne soit mise en place).

Dans une première phase, le processus *Intergalactique* traitera *séquentiellement* chaque serveur central. Dans une deuxième phase, on *parallélisera* au maximum les requêtes. La première phase doit être pensée et conçue sur le papier afin de pouvoir facilement passer à la deuxième phase. La première phase peut suffire pour passer à la

suite du projet, mais elle est obligatoire et devra être rendue. On conservera toutes les versions des différentes phases du programme à des fins de test.

3.5.2 Raccordement

On créera un programme *Gestion* qui devra accepter sur sa ligne de commande un fichier de configuration dans lequel on trouvera les informations suivantes :

- la taille de la mémoire de gestion des sms d’un central
- le nom du fichier *annuaire des serveurs* ;
- le nom d’un fichier *annuaire global des téléphones* ;

Le processus *Gestion* devra créer les tuyaux, effectuer les clônages et les recouvrements nécessaires afin de créer l’ensemble des processus nécessaires à la simulation.

4 Évolutions complémentaires et optionnelles

Voici une liste d’extensions possibles, de la plus simple à la plus compliquée. Cette liste est non-limitative.

4.1 Identification des sms

Chaque sms a un numéro unique l’identifiant par téléphone. Si la simulation tourne très longtemps, ce numéro risque de dépasser la taille du plus grand entier disponible en langage C. Une solution serait d’utiliser une liste contenant un nombre fini d’identifiants uniques, chaque sms piochant dans cette liste. Si votre téléphone est découpé en plusieurs threads, quels problèmes cela pose-t-il ? Formalisez le problème, étudiez une solution, et implémentez cette solution.

4.2 Des numéros de téléphones inconnus

Que se passe-t-il si on décide que les téléphones peuvent émettre vers des numéros qui n’existent pas ? Il faut que quelqu’un décide que le sms ne peut être acheminé. Dans notre architecture, les serveurs centraux et le serveur intergalactique connaissent tous les numéros existants, mais dans la réalité, ce n’est pas forcément le cas. Discutez quel est le serveur le mieux adapté pour ce rôle (avantages, inconvénients), modélisez la proposition qui vous semble la plus adaptée, et implémentez-la.

4.3 Des annuaires partiels

Dans l’approche actuelle du problème, tant les téléphones, que les serveurs centraux que le serveur intergalactique connaissent tout l’annuaire de tous les téléphones. Comment-faire si les serveurs centraux ne connaissent que les téléphones qui sont dans leur réseau ? Plusieurs approches sont possibles, certaines très simples, d’autres très compliquées. Discutez plusieurs approches, et modélisez et implémentez celle qui vous semble la meilleure au sens d’un critère adapté (la plus réaliste, la plus simple, etc.)

4.4 Utilisation de socket

L'utilisation de communications entre machines ne fait pas partie des objectifs de ce cours. Cependant, afin de rendre le projet plus vivant et plus attractif, nous proposons aux plus débrouillards d'utiliser des socket de sorte que la simulation soit plus réaliste :

- chaque telephone communique avec son serveur central par des socket ;
- chaque serveur central communique avec le réseau intergalactique par socket.

Les informations nécessaires à l'utilisation des socket peuvent facilement se trouver sur internet.

Le travail de mise en œuvre des communications par socket doit être entrepris **uniquement après avoir réussi à programmer une application complètement opérationnelle en mode texte.**

Attention : **ne jamais modifier un programme qui fonctionne et toujours travailler sur une copie de celui-ci.**

5 Annexes

5.1 De l'intérêt des standards pour assurer une meilleure interopérabilité

Le recours à des protocoles "standardisés" (ou pour le moins communs) à un double intérêt :

- il permet de gagner du temps sur le développement de ce projet et d'illustrer par la pratique la façon dont les problèmes sont traditionnellement résolus en informatique :
- il permettra de mélanger les projets entre eux afin de tester le respect du protocole et peut-être, d'aider certains binômes à avancer (il suffira d'utiliser les processus d'un autre binôme³).

La capacité ainsi esquissée à mélanger des composants issus de programmeurs ou prestataires différents pour constituer un système d'information unique se nomme "interopérabilité". C'est un des enjeux majeurs de l'informatique actuelle.

5.2 Spécifications de protocoles de communication

Les messages sont constitués de caractères ASCII (sans accent). Chaque message est constitué de différents champs séparés par les caractères '|' et terminés par un caractère de fin de message '\n'. Le premier caractère indique toujours la nature du message.

Le récapitulatif des échanges est présenté sur la figure 3.

Remarque : On considérera que le sms ne contient pas le caractère "\n".

5.2.1 Un format de message unique

Quelque soit le message, son format est toujours le même :

³Bien entendu, cette utilisation ne peut s'entendre qu'à des fins de mise au point...

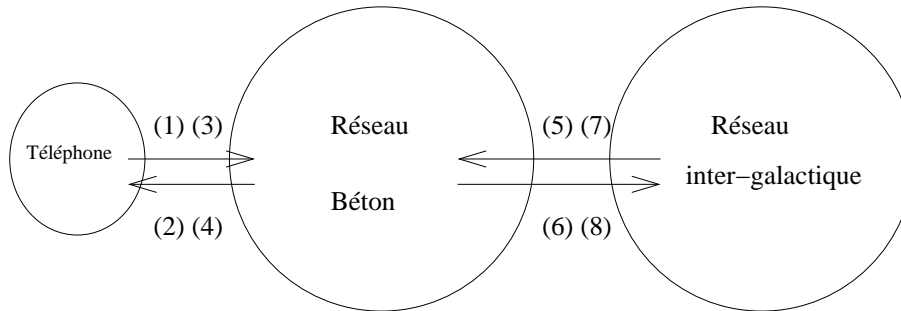


FIG. 3 – Récapitulatif des échanges

|E...E|D..D|C...C|A...A|B...B\n

avec :

- E . . . E est l'identifiant de l'envoyeur du message (longueur strictement supérieur à 0 et indéterminé *a priori*) ;
- D . . . D est l'identifiant du destinataire du message (longueur strictement supérieur à 0 et indéterminé *a priori*) ;
- C . . . C est le nom de la commande décrite dans le message (longueur strictement supérieur à 3 et indéterminé *a priori*) ;
- A . . . A est le premier argument (optionnel) de la commande (longueur indéterminée *a priori*) ;
- B . . . B est le deuxième argument (optionnel) de la commande (longueur indéterminée *a priori*) ;

Ce format devrait permettre de traiter l'ensemble des cas possibles. Afin de simplifier la mise en œuvre, on supposera qu'un message comporte au plus 255 caractères.

La suite des spécifications du protocole consiste à définir l'utilisation de ce format spécifiquement au contexte des communications énumérées dans le schéma récapitulatif 3.

5.2.2 Protocole sur le réseau

Les échanges utilisent deux types de message, l'*émission de sms* (identique à la *réception de sms*), et l'*accusé de réception*.

(1)(2)(6)(7) *Message de sms (émission et réception)*

Ce message est envoyé par le téléphone et réceptionné par le serveur central, ou est envoyé par le serveur central et est réceptionné par le téléphone. Il circule également entre les serveurs centraux au travers du réseau intergalactique.

Format : |E . . . E|D . . . D|SMS|A . . . A|B . . . B\n

Chaque lettre représente un caractère du message.

- E . . . E est l'identifiant de l'émetteur du sms (par exemple son numéro de téléphone) ;

- D...D est l'identifiant du destinataire du sms (par exemple son numéro de téléphone);
- A...A correspond à un numéro *unique* identifiant le sms sur le téléphone;
- B...B correspond au message sms lui même, une suite de caractère alphanumériques;

(3)(4)(5)(8) *Message d'accusé de réception*

Ce message est soit envoyé par un téléphone, soit émis par un serveur central au bout d'un certain nombre d'essais. Il circule sur les serveurs centraux, le réseau intergalactique, pour être finalement réceptionné par un téléphone.

Format : |E...E|D...D|ACCUSE|A...A|B...B\n avec :

- E...E est l'identifiant de l'émetteur *initial* du sms (par exemple son numéro de téléphone);
- D...D est l'identifiant du destinataire *initial* du sms (par exemple son numéro de téléphone);
- A...A est le numéro identifiant le sms initialement envoyé;
- B...B est la réponse :
 - `recu` si le sms a été reçu;
 - `timeout` si le délai imparti pour la réceptionner la réponse est écoulé;
 - `probleme` si un problème inattendu est survenu.

5.3 Générateur aléatoire et manipulation de chaînes en C

Pour un exemple de générateur aléatoire, vous pouvez télécharger le lien suivant

<http://www.esiee.fr/%7Einfo/a2si/TC-ESIEE/IN301ST01/chaine.tz>

Pour le décompresser, tapez la commande `'tar zxvf chaine.tz'`.

Le code source contient également des exemples de manipulations de chaînes de caractères en C, utiles pour découper et créer les messages, c'est-à-dire pour manipuler le protocole décrit plus haut.

5.4 Redirection des entrées et des sorties

La redirection des entrées et des sorties peut se faire grâce à l'appel système `dup`.

```
int dup2(int oldfiledes , int newfiledes);
```

Cet appel système sert à dupliquer le contenu du descripteur `oldfiledes` dans un autre descripteur `newfiledes`. Il renvoie -1 en cas d'échec, et il prend en argument deux descripteurs de fichiers. Par exemple si on a envie que `STDIN` (`==1`) soit en fait `STDERR` (`==2`) on peut écrire ceci :

```
dup2(2 , 1);
```

L'exemple suivant redirige l'entrée et la sortie standard vers un tube, et recouvre le processus courant et son fils par deux programmes. Le père et le fils vont communiquer par leur entrée et sortie standard.

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

void usage(char * basename) {
    fprintf(stderr,
        "usage: %s [<programme_1> [<programme_2>]]\n",
        basename);
    exit(1);
}

int main(int argc, char *argv[]) {
    int pid; /* permet d'identifier qui on est*/
    int fdpipe[2]; /* sera utilisé pour lier les processus */

    if (argc != 3) usage(argv[0]);

    /* on créé le pipe qui sera utilisé pour relier
       la sortie du premier processus
       vers l'entrée du second
    */
    if ( pipe(fdpipe) == -1 ) {
        perror("pipe");
        exit(-1);
    }

    switch(pid = fork()) {
        case -1:
            /* le fork a échoué */
            perror("fork");
            exit(-1);
        case 0:
            /* code du fils */
            /* on fait en sorte que lorsque le processus
               écrira sur l'entrée standard (1)
               il le fera en fait dans le pipe (fdpipe[1])
            */
            dup2(fdpipe[1], 1);
            /* on ferme tout, même le pipe...
               on n'en a plus besoin
            */
            close(fdpipe[0]);
            close(fdpipe[1]);
            execlp(argv[1], argv[1], NULL);
            /* pas besoin de break,
               ce code n'existe déjà plus à l'exécution
            */
        default :
            /* code du père */
            /* on fait en sorte que lorsque le processus

```

```
        lira sur la sortie standard (0)  
        il le fera en fait dans le pipe (fdpipe[0])  
    */  
    dup2(fdpipe[0], 0);  
    close(fdpipe[0]);  
    close(fdpipe[1]);  
    execlp(argv[2], argv[2], NULL);  
    }  
    /*  
    cette portion de code ne sera jamais exécutée,  
    puisque les processus ont déjà  
    été remplacé. On met néanmoins un  
    return sinon le compilateur proteste.  
    */  
    return 0;  
}
```