

Simulation d'un système de paiement par carte bancaire

Mini projet IN301/IN3ST01 - 2006-2007

D'après un projet de Jérôme Gueydan pour l'ENSTA

15 novembre 2006

Table des matières

1	Introduction	4
1.1	Le principe du paiement par carte bancaire	4
1.2	La demande d'autorisation	5
1.3	Le routage	6
2	Cahier des charges	6
2.1	Cahier des charges fonctionnelles	7
2.1.1	Architecture fonctionnelle	7
2.1.2	Le terminal	8
2.1.3	Le serveur d'acquisition	8
2.1.4	Le serveur d'autorisation	9
2.1.5	Le réseau interbancaire	9
2.2	Cahier des charges techniques	9
2.2.1	Contraintes générales	9
2.2.2	Nombre de processus	9
2.2.3	Paramètres	9
2.2.4	Gestion des échanges par tuyaux	10
2.3	Comment tester sans s'y perdre ?	10
2.4	Livrables	10
3	Conduite du projet	11
3.1	Introduction	11
3.1.1	Un projet en plusieurs étapes	11
3.1.2	Méthode "diviser pour régner"	11
3.2	Étape 1 : les fonctions de communication	12
3.3	Étape 2 : réalisation de programmes indépendants	12
3.3.1	Processus : <i>Autorisation</i>	12

3.3.2	Processus : <i>Acquisition</i>	12
3.3.3	Processus : <i>Terminal</i>	13
3.4	Etape 3 : création du réseau bancaire	13
3.4.1	Préparation de la communication par tuyaux	13
3.4.2	Raccordement	13
3.5	Etape 4 : création du réseau interbancaire	14
3.5.1	Processus : <i>Interbancaire</i>	14
3.5.2	Raccordement	14
4	Évolutions complémentaires et optionnelles	14
4.1	Cumul des transactions	14
4.2	Délai d'annulation	15
4.3	Utilisation de socket	15
5	Annexes	15
5.1	De l'intérêt des standards pour assurer une meilleure interopérabilité	15
5.2	Spécifications de protocoles de communication	16
5.2.1	Un format de message unique	16
5.2.2	Protocole du terminal	17
5.2.3	Protocole du réseau interbancaire	17
5.2.4	Le serveur d'autorisation	18
5.3	Générateur aléatoire	18
5.4	Redirection des entrées et des sorties	18

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 2007, à une date qui sera précisée ultérieurement.

Pour faire court, sachez que tous les projets "semblables" auront une note de zéro, 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>.

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 2007 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 doit 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 tournant 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 du rapport, et de la qualité du 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 fait. 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 aurez écrit n'est pas proprement justifié.

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

Remarque

L'énoncé peut sembler épais de prime abord. Ce n'est pas le cas : il se résume aux pages 4 à 10. 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.

Ce projet est repris de celui proposé par Jérôme Gueydan

http://www.ensta.fr/gueydan/IMG/pdf/IN201_projet_2008.pdf

Les deux projets ne sont cependant pas strictement identiques.

1 Introduction

L'objectif du projet est de simuler les échanges entre banques permettant à un particulier de payer ses achats avec sa carte bancaire (dite "carte bleue"), même si celle-ci n'est pas émise par la même banque que celle du vendeur.

Avant de présenter le sujet, examinons le fonctionnement du paiement par carte bancaire.

1.1 Le principe du paiement par carte bancaire

Le paiement par carte bancaire met en relation plusieurs acteurs :

- le client, qui souhaite régler un achat avec la carte bancaire qu'il possède et qui lui a été fournie par sa banque (le Crédit Chaton) ;
- le commerçant, qui est équipé d'un terminal de paiement fourni par sa propre banque (la Bénépé) ;
- la banque du commerçant (la Bénépé) à laquelle est connectée le terminal de paiement ;
- la banque du client (le Crédit Chaton) qui va dire si la transaction est autorisée (donc si le compte de son client est suffisamment provisionné ou non).

Le terminal du commerçant est relié à la banque Bénépé grâce à une simple ligne téléphonique. La banque Bénépé est connectée à toutes les autres banques installées en France, et notamment au Crédit Chaton, grâce à un réseau dédié : le réseau interbancaire (voir fig. 1).

Supposons maintenant que le client lambda se rend chez son revendeur de logiciels préféré pour acheter la toute dernière version du système d'exploitation "FENETRES". Au moment de passer en caisse, il dégage sa carte bancaire, le caissier l'insère dans son terminal de paiement et le client doit, après avoir regardé au passage la somme qu'il s'apprête à déboursier, saisir son code confidentiel.

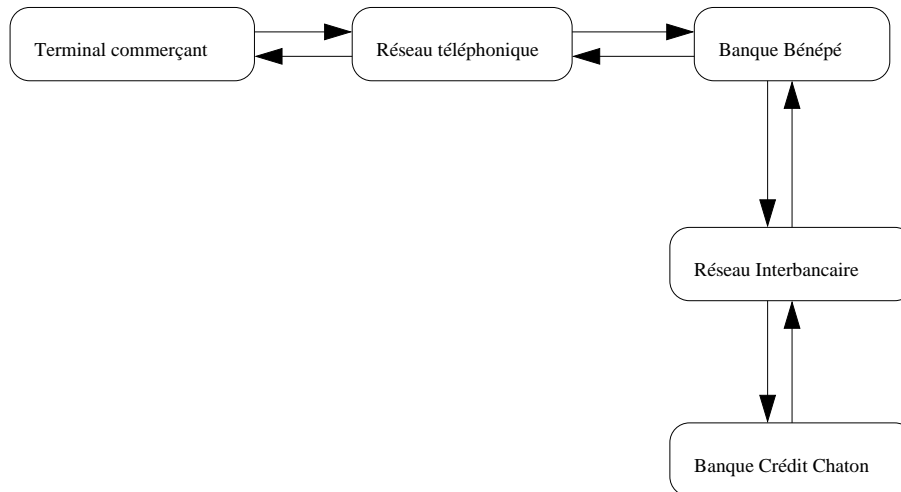


FIG. 1 – Principe de communication des téléphones et du central

Ce code est directement vérifié par la carte (plus exactement par la puce contenue dans la carte). Si le code est erroné, la transaction s’arrête là. Si le code est bon, en revanche, les opérations suivantes ont lieu :

1. Le terminal se connecte (via le réseau téléphonique) au serveur de la banque Bénépé et envoie le numéro de la carte bancaire ainsi que le montant de la transaction.
2. Le serveur de la banque Bénépé regarde le numéro de la carte et, se rendant compte qu’il ne s’agit pas d’une des cartes qu’il a émises, envoie le numéro de carte avec le montant de la transaction au serveur de la banque Crédit Chaton, via le réseau interbancaire permettant de relier les différentes banques.
3. Le serveur de la banque Crédit Chaton prend connaissance du numéro de la carte bancaire et vérifie que le compte correspondant à ce numéro dispose d’un solde suffisant pour honorer la transaction.
4. Si c’est le cas, il répond à la banque Bénépé (toujours via le réseau interbancaire) que le paiement est autorisé. Si ce n’est pas le cas, il répond le contraire.
5. Enfin, le serveur de la banque Bénépé transmet la réponse au terminal du commerçant.
6. La transaction est validée (“paiement autorisé”) ou refusé (“paiement non autorisé”).

1.2 La demande d’autorisation

La suite des opérations décrites ci-dessus se nomme la “demande d’autorisation” et a essentiellement pour but de vérifier que le compte client est bien provisionné (ou

qu'il a une autorisation de découvert). Cette demande d'autorisation n'est pas systématique et dépend du terminal¹, lequel prend par exemple en compte le montant de la transaction.

La demande d'autorisation transite via deux serveurs différents :

Le serveur d'acquisition - Il s'agit du serveur de la banque du commerçant auquel se connecte le terminal via le réseau téléphonique. Une fois connecté, le terminal envoie au serveur d'acquisition toutes les informations concernant la transaction, notamment le montant, le numéro de carte et des données permettant d'assurer la sécurité de la transaction.

Le serveur d'autorisation - Il s'agit du serveur de la banque du client auquel le serveur d'acquisition transmet l'autorisation de paiement émise par le terminal. La réponse à la demande suit le chemin inverse, à savoir serveur d'autorisation de la banque du client → serveur d'acquisition de la banque du commerçant → terminal du commerçant.

1.3 Le routage

Pour effectuer le routage des demandes d'autorisation, c'est-à-dire pour déterminer à quelle banque chaque demande d'autorisation doit être transmise, le serveur d'acquisition utilise les premiers numéros de chaque carte bancaire concernée : ceux-ci indiquent la banque ayant émis cette carte. Dans ce projet, nous partirons des principes suivants :

- un numéro de carte est constitué de seize chiffres décimaux ;
- les quatre premiers correspondent à un code spécifique à chaque banque ;
- les serveurs d'acquisition des banques sont directement reliés au réseau interbancaire.

Chaque serveur d'acquisition analyse donc le numéro de la carte qui figure dans la demande d'autorisation qu'il reçoit, puis :

- si le client est dans la même banque que le commerçant (et que le serveur d'acquisition), il envoie la demande directement au serveur d'acquisition de cette banque ;
- si le client est dans une autre banque, le serveur d'acquisition envoie la demande sur le réseau interbancaire, sans se préoccuper de la suite du transit.

Le réseau interbancaire n'est donc pas un simple réseau physique : il doit aussi effectuer le routage des demandes d'autorisation, c'est-à-dire analyser les demandes qui lui sont fournies, envoyer chaque demande vers le serveur d'autorisation de la banque correspondante 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 terminal envoyant une demande d'autorisation au serveur d'acquisition de sa banque ;

¹et bientôt aussi des cartes bancaires.

- le serveur d’acquisition effectuant le routage de la transaction vers le bon serveur d’autorisation et effectuant le routage des réponses qu’il reçoit en retour des terminaux ;
- le réseau interbancaire auquel sont connectés les différents serveurs d’acquisition, capable d’effectuer le routage des demandes et des réponses relayées par les serveurs d’acquisition ;
- le serveur d’autorisation fournissant la réponse à la demande d’autorisation ;
- le tout permettant un maximum de parallélisme.

Remarque : 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 des systèmes monétiques, afin de valider leur fonctionnement avant leur mise en production.

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.

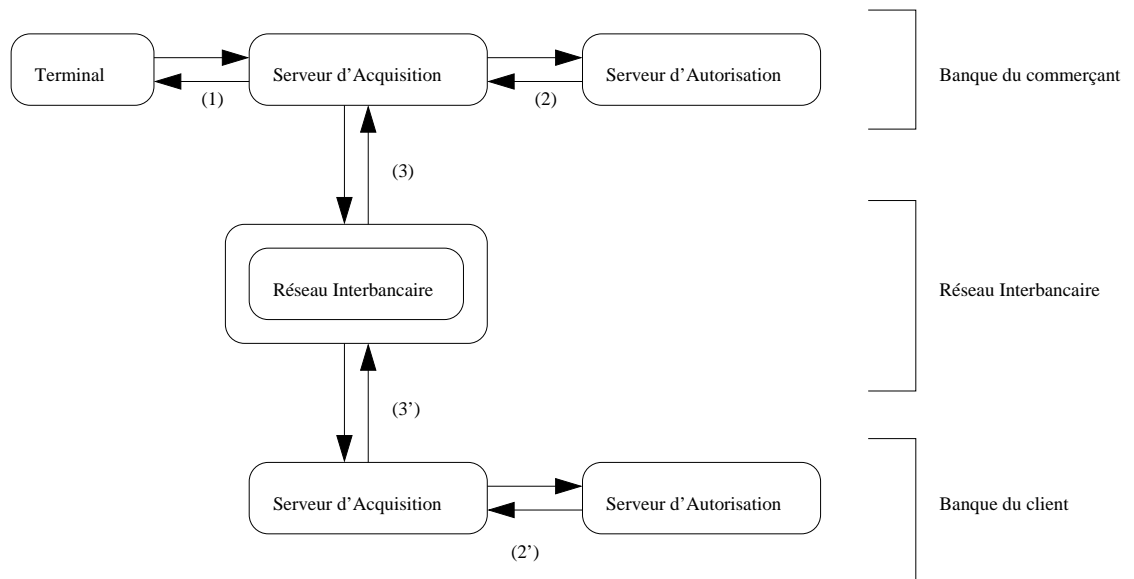


FIG. 2 – Architecture fonctionnelle du projet

Le terminal est relié via le réseau téléphonique (1) au serveur d’acquisition de la

banque du commerçant. Celui-ci est connecté au sein de la banque (2) au serveur d'autorisation de cette même banque.

Le réseau interbancaire relie (3,3') les serveurs d'acquisition des différentes banques. Toutes les autres banques de la place sont également reliées au réseau interbancaire, mais ne sont pas représentées sur ce schéma.

2.1.2 Le terminal

Dans le cadre de ce projet, il n'est pas question d'utiliser de vrais terminaux ou de vraies cartes. Le terminal étant un moyen d'envoyer aux programmes des demandes d'autorisation, il sera simulé pour ce projet par un exécutable qui enverra un numéro de carte aléatoirement tirée dans la liste des cartes existantes et un montant de transaction aléatoire.

Pour fonctionner, un terminal a donc besoin de connaître la liste des cartes existantes.

Chaque terminal devra envoyer les informations générées au serveur d'acquisition, devra attendre la réponse du serveur d'acquisition et affichera cette réponse à l'écran (*i.e.* paiement autorisé ou non).

Les échanges entre le terminal et le serveur d'acquisition ont lieu suivant un protocole bien déterminé : les informations sont formatées d'une certaine façon. Ce sont les constructeurs de terminaux qui imposent leurs protocoles et ce sont les serveurs d'acquisition qui doivent s'adapter pour "parler" les protocoles des différents terminaux qui lui sont connectés.

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 Le serveur d'acquisition

Le serveur d'acquisition n'a qu'une fonction de routage :

- il doit pouvoir accepter des demandes d'autorisation provenant de terminaux et du réseau interbancaire ;
- il doit pouvoir effectuer le routage des demandes d'autorisation vers le serveur d'autorisation de la banque ou bien vers le réseau interbancaire ;
- il doit pouvoir accepter les réponses provenant du réseau interbancaire ou du serveur d'autorisation de la banque ;
- il doit pouvoir envoyer les réponses vers le réseau interbancaire ou le terminal (en étant capable d'apparier chaque réponse à la demande initiale).

Le serveur d'acquisition doit être capable d'utiliser le protocole de communication employé par les terminaux, ainsi que le protocole du réseau interbancaire et le protocole du serveur d'autorisation (voir les protocoles définis en annexe).

Afin de pouvoir effectuer correctement le routage des messages, le serveur d'acquisition doit connaître les 4 premiers chiffres des numéros des cartes de sa banque.

2.1.4 Le serveur d'autorisation

Le serveur d'autorisation doit être capable de fournir une réponse à une demande d'autorisation. Pour fonctionner, le serveur d'autorisation doit donc avoir accès aux soldes des comptes des clients de la banque référencés par numéro de carte.

Dans le cadre de ce projet, nous utiliserons une méthode simple : le serveur d'autorisation possède la liste des numéros de cartes émises par la banque, auxquels sont associés les soldes des comptes adossés à ces cartes ; lorsqu'une demande d'autorisation lui parvient, le serveur vérifie que le numéro de carte figure bien dans sa liste. Il contrôle alors que le solde de compte est suffisant pour effectuer la transaction, si c'est le cas il répond oui, sinon il répond non.

2.1.5 Le réseau interbancaire

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

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

Pour fonctionner, le réseau interbancaire doit posséder la liste des codes de 4 chiffres figurant en entête des cartes et les banques associées.

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 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 terminal (que l'on nommera *Terminal*), un processus pour le serveur d'acquisition (*Acquisition*), un processus pour le serveur d'autorisation (*Autorisation*) et un processus pour le réseau interbancaire (*Interbancaire*).

La simulation pouvant mettre en jeu plusieurs terminaux, plusieurs banques, etc., on trouvera en fait un processus *Terminal* par terminal, un processus *Acquisition* par serveur d'acquisition et un processus *Autorisation* par serveur d'autorisation.

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

2.2.3 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 :

- la liste des banques et de leur code à 4 chiffres ;
- la liste des cartes bancaires de chaque banque et le solde du compte associé.

Ces fichiers seront au format texte, chaque ligne contenant les deux informations demandées (code sur 4 chiffres et banque associé ou numéro de carte et solde du compte associé), séparées par un espace.

2.2.4 Gestion des échanges par tuyaux

Les terminaux connectés au même serveur d'acquisition (donc les terminaux d'une même banque) utiliseront chacun une paire de tuyaux pour dialoguer avec ce serveur. Le serveur d'acquisition aura donc comme rôle d'orchestrer simultanément les lectures et les écritures sur ces tuyaux.

Les échanges entre un serveur d'acquisition et un serveur d'autorisation seront possibles au travers d'une paire de tuyaux fonctionnant d'une façon très classique.

Pour ce qui concerne le réseau interbancaire et les serveurs d'acquisition, la problématique est strictement identique à celle des terminaux : il faudra utiliser une paire de tuyaux pour connecter chaque serveur d'acquisition au réseau interbancaire. Ceci confère au processus *Interbancaire* un rôle de chef d'orchestre et implique de maintenir une table de routage.

2.3 Comment tester sans s'y perdre ?

Le schéma proposé ci-dessus comporte un petit défaut tant que les communications entre processus se feront sur la même machine (donc sans utiliser de *socket*, développement proposé en option) : chaque *Terminal* 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 *Terminal* d'une entrée et d'une sortie standard qui lui soient propres, une astuce peut être utilisée : lors de la création d'un nouveau *Terminal* par le processus *Acquisition*, recouvrir le nouveau fils du processus non pas par *Terminal*, mais par *xterm -e Terminal*.

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 fichiers 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 "sereine" de la programmation de ce projet.

3.1 Introduction

3.1.1 Un projet en plusieurs étapes

Le développement de ce simulateur qui est présenté ici est 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 particulièrement 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 garantiront la réussite de ce projet et simplifient grandement son développement.

3.2 Etape 1 : les fonctions de communication

La première étape consiste à programmer les différentes fonctions de communication permettant de lire et écrire les demandes d'autorisation et des réponses 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.

Une fois ces fonctions écrites, elles seront 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 : *Autorisation*

Le processus *Autorisation* recevra sur son entrée standard des demandes d'autorisation auxquels il devra répondre sur sa sortie standard.

L'exécutable *Autorisation* prendra sur sa ligne de commande le nom de fichier de paramètres, c'est-à-dire celui comprenant la liste des soldes des comptes des clients de la banque, référencés par leur numéros de cartes.

Le processus *Autorisation* pourra donc être testé indépendamment des autres processus.

3.3.2 Processus : *Acquisition*

Le processus *Acquisition* recevra des messages de demande d'autorisation en provenance soit des terminaux, soit du réseau interbancaire, ou bien en provenance du serveur d'autorisation ou du réseau interbancaire. Ces messages seront redirigés ("routés") vers le réseau interbancaire, vers le serveur d'acquisition ou bien vers les terminaux, 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 d'autorisation ("à la main") ;
- des fichiers dans lesquels le processus *Acquisition* ira écrire lorsqu'il est supposé envoyer un message ;
- des fichiers dans lesquels le processus *Acquisition* 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 *Terminal* et *Interbancaire*.

Le programme devra accepter sur sa ligne de commande au moins un paramètre représentant les 4 premiers chiffres des cartes de la banque à laquelle il appartient.

Dans une première phase, le processus *Acquisition* traitera *séquentiellement* chaque terminal ainsi que les messages en provenance ou à destination du serveur d'autorisation et du réseau interbancaire. 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.

3.3.3 Processus : *Terminal*

Le processus *Terminal* offrira l'interface permettant d'envoyer et de recevoir les informations pour une transaction (montant et numéro de carte).

Afin de pouvoir tester le fonctionnement du *Terminal*, les messages de demande d'information à destination du serveur d'acquisition pourront être écrits dans un fichier. Les réponses seront également lues dans un fichier qui aura été préparé à l'avance.

3.4 Etape 3 : création du réseau bancaire

3.4.1 Préparation de la communication par tuyaux

En pratique, le processus *Acquisition* et chaque processus *Terminal* vont communiquer par une paire unique de tuyaux. Une fois ces deux tuyaux créés, il est possible de modifier simplement les programmes *Terminal* et *Acquisition* pour qu'ils utilisent non pas des fichiers mais ces deux tuyaux.

Pour cela, on complétera le programme *Terminal* en permettant de lui passer sur la ligne de commande les descripteurs des tuyaux à utiliser en lecture et en écriture. On modifiera le code pour que ces tuyaux soient utilisés en lieu et place des fichiers employés à la deuxième étape.

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 *Acquisition* et chaque *Terminal* (il y aura autant de paires de tuyaux que de terminaux, et d'autre part, une paire de tuyaux entre *Acquisition* et *Autorisation*, après avoir redirigé leurs entrées et sorties standards (voir annexe).

Ecrire un programme *Banque* acceptant sur sa ligne de commande le nombre de terminaux de la banque. Le programme devra également accepter sur sa ligne de commande les paramètres suivants :

- le nom de la banque à simuler ;
- les 4 chiffres associés à cette banque ;
- le nom d'un fichier contenant les soldes des comptes clients ;
- le nombre de terminaux à créer.

Créer les tuyaux nécessaires, opérer les clonages et recouvrements nécessaires pour créer les processus *Terminal* et *Autorisation* en nombre suffisant.

Enfin, recouvrir *Banque* par le programme *Acquisition* sans oublier les redirections nécessaires des tuyaux avec les entrées et sorties standards.

3.5 Étape 4 : création du réseau interbancaire

3.5.1 Processus : *Interbancaire*

Chaque serveur d'acquisition sera relié au processus *Interbancaire* par une paire de tuyaux. Celle-ci permettra à *Interbancaire* 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 *Interbancaire* et les différents processus *Acquisition* sera similaire à celle mise en place entre chaque processus *Acquisition* et les processus *Terminal* qui y sont reliés.

Dans un premier temps, les messages transitant par *Interbancaire* 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 *Interbancaire* traitera *séquentiellement* chaque serveur d'acquisition. Dans une deuxième phase, on *parralélisera* 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 *Démarrage* qui devra accepter sur sa ligne de commande un fichier de configuration dans lequel on trouvera les informations suivantes :

- le nom d'un fichier contenant toutes les banques et les 4 chiffres associés à chaque banque ;
- les noms des fichiers nécessaires au fonctionnement de chaque banque ;
- le nombre de terminaux pour chaque banque.

Le processus *Démarrage* devra créer les tuyaux, effectuer les clonages et les recouvrements nécessaires afin de créer l'ensemble des processus nécessaires à la simulation.

4 Évolutions complémentaires et optionnelles

4.1 Cumul des transactions

On demande de rendre la simulation plus réaliste en permettant au niveau du serveur d'autorisation de comptabiliser l'ensemble des paiements effectués par une carte, afin de proposer une réponse à la demande d'autorisation qui tiennent compte du solde du compte et de l'encours de la carte.

4.2 Délai d'annulation

Dans cette version, les processus *Acquisition* et *Interbancaire* doivent pouvoir gérer une fonction d'annulation : une transaction est annulée lorsqu'il s'est écoulé plus de 2 secondes depuis le relai de la demande, sans qu'une réponse ne soit parvenue. Si une réponse parvient ultérieurement, elle sera ignorée.

Ceci suppose que les processus *Acquisition* et *Interbancaire* conservent une trace datée de toutes les demandes qu'ils traitent et qu'ils vérifient régulièrement la date de péremption de chaque demande.

Attention : l'annulation d'une demande pour délai trop important ne dispense pas les processus *Acquisition* et *Interbancaire* d'envoyer une réponse à l'émetteur de la demande.

La synchronisation (demande, autorisation d'écriture, ordre de lecture) sera mise en œuvre par l'intermédiaire de l'appel système `select()`.

4.3 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 terminal communique avec son serveur d'acquisition par des `socket` ;
- chaque serveur d'acquisition communique avec le réseau interbancaire 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 binôme²).

²Bien entendu, cette utilisation ne peut s'entendre qu'à des fins de mise au point...

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.

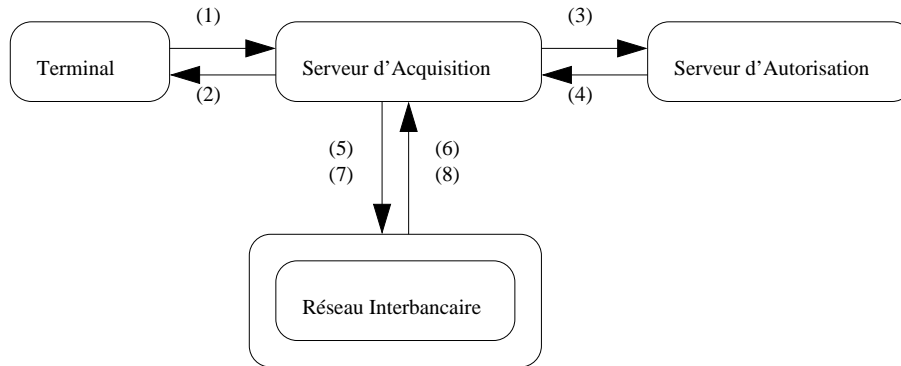


FIG. 3 – Récapitulatif des échanges

Remarque : afin de simplifier au maximum le protocole, on considère qu'il n'est pas possible que plus d'une demande d'autorisation ou réponse, correspondant à une transaction faite avec une carte donnée, circule sur le réseau. Cette hypothèse, assez réaliste, permettra de simplifier l'appariement des demandes et des réponses au niveau des processus *Acquisition* et *Interbancaire*.

5.2.1 Un format de message unique

Quelque soit le message, son format est toujours le même : |I...I|C...C|A...A|B...B|\n avec :

- I . . . I est l'identifiant de l'envoyeur 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 9 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 énumérés dans le schéma récapitulatif 3.

5.2.2 Protocole du terminal

Les échanges avec le terminal utilisent deux types de message, *la demande d'autorisation* et *la réponse à la demande d'autorisation*.

(1) Message de demande d'autorisation

Ce message est envoyé par le terminal et réceptionné par le serveur d'acquisition.

Format : |I...I|DemandeAutorisation|A...A|B...B|\n

Chaque lettre représente un caractère du message.

- I...I est l'identifiant du terminal (par exemple son pid ;
- A...A correspond au montant de la transaction exprimé en centimes d'euros ;
- B...B correspond au numéro de la carte sur 16 chiffres décimaux (longueur constante).

(2) Message de réponse à la demande d'autorisation

Ce message est envoyé par le serveur d'acquisition et réceptionné par le terminal.

Format : |I...I|ReponseAutorisation|A...A|\n avec :

- I...I est l'identifiant du serveur d'acquisition (par exemple son pid ;
- A...A est la chaîne de caractère :
 - oui si la demande est acceptée ;
 - non si la demande est refusée ;
 - timeout si le délai imparti pour la réceptionner la réponse est écoulé ;
 - probleme si un problème inattendu est survenu.

5.2.3 Protocole du réseau interbancaire

(5)(7) Message de demande d'autorisation

Ce message peut être émis par le réseau interbancaire et réceptionné par un serveur d'acquisition ou bien émis par un serveur d'acquisition et réceptionné par le réseau interbancaire.

Format : |I...I|DemandeAutorisation|A...A|B...B|\n avec :

- I...I est l'identifiant de l'expéditeur du message (peut être son PID) ;
- A...A correspond au montant de la transaction exprimé en centimes d'euros ;
- B...B correspond au numéro de la carte sur 16 chiffres décimaux (longueur constante).

(6)(8) Message de réponse

Ce message est fourni par un serveur d'acquisition au réseau interbancaire ou par le réseau interbancaire au serveur d'acquisition.

Format : |I...I|ReponseAutorisation|A...A|B...B|\n avec

- I...I est l'identifiant de l'expéditeur du message (peut être son PID) ;
- A...A est la chaîne de caractère :
 - oui si la demande est acceptée ;
 - non si la demande est refusée ;

- timeout si le délai imparti pour réceptionner la réponse est écoulé ;
- probleme si un problème inattendu est survenu.
- B...B est le numéro de carte (16 chiffres) sur laquelle la transaction a été passée.

5.2.4 Le serveur d'autorisation

Message de demande d'autorisation

Ce message est envoyé par le serveur d'acquisition au serveur d'autorisation.

Format : |I...I|DemandeAutorisation|A...A|B...B|\n avec :

- I...I est l'identifiant du serveur d'acquisition (peut être son PID) ;
- A...A correspond au montant de la transaction exprimé en centimes d'euros ;
- B...B correspond au numéro de la carte sur 16 chiffres décimaux (longueur constante).

Message de réponse

Ce message est envoyé par le serveur d'autorisation au serveur d'acquisition.

Format : |I...I|ReponseAutorisation|A...A|B...B|\n avec :

- I...I est l'identifiant du serveur d'autorisation (peut être son PID) ;
- A...A est la chaîne de caractère :
 - oui si la demande est acceptée ;
 - non si la demande est refusée ;
 - timeout si le délai imparti pour réceptionner la réponse est écoulé ;
 - probleme si un problème inattendu est survenu.
- B...B est le numéro de carte (16 chiffres) sur laquelle la transaction a été passée.

5.3 Générateur aléatoire

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'.

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 != 2) 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;
}
```