OCM2000

Carte à puce internet & Objets mobiles embarqués

Pascal Urien - Bull CP8 R&D
68 routes de Versailles BP 45 78431 Louveciennes Cedex - France
Pascal.Urien@Bull.net - Tel: +33 1 39 66 42 30
OCM2000 Objects, Components, Models "Past, Present, Future" - May 18, 2000

1. Résumé.

Ce papier présente une architecture novatrice de carte à puce internet (baptisée *Oversoft*), et dresse un panorama des possibles nouvelles architecture de sécurité, à base d'objets embarqués.

2. Introduction

Depuis 1998 Bull CP8 mène des recherches sur une carte à puce spécifiquement dédiée au réseau internet. L'idée de base est de considérer la carte comme un nœud actif du réseau (*network computer)* capable d'utiliser les ressources du terminal auquel elle est reliée (clavier - écran - souris - navigateur - accès internet). Nous pensons qu'une carte à puce internet est un bon vecteur pour le commerce d'objet virtuel à travers internet, et que les ressources offertes par la carte pourraient être mise en œuvre grâce à une architecture objet ouverte et répartie.

3. La carte à puce classique.

La carte à puce à microprocesseur (*SPOM* - **S**elf **P**rogrammable **O**ne-chip **M**icrocomputer) a été inventée à la fin des années 70 par Michel Ugon [a2]. Les puces actuelles, dont la surface n'excède pas 25 mm^2 [a4] sont principalement à base de processeurs 8 bits 6805 ou 8051, qui comportent quelques dizaines de Kilo-octets de ROM/EEPROM et quelques centaines d'octets de RAM. La communication avec le monde extérieur est assurée par une liaison série half duplex dont le débit est compris entre 9600 et 230400 bauds. La puissance des processeurs actuels est de l'ordre du Mips, mais de nouveaux composants RISC 32 bits permettrons d'obtenir des puissances de traitement de l'ordre de 50 Mips [a8]. Les normes ISO 7816 [n1,n2,n3,n4] décrivent les protocoles de communications mises en œuvre par le SPOM, T=0 est un protocole d'échange en mode caractères, T=1 est un protocoles d'échanges utilisant des blocs munis d'un CRC. La puce réalise schématiquement deux types de services :

- Des algorithmes cryptographiques utilisés pour le chiffrement, l'authentification, la génération de certificat.
- Le stockage de données qui sont protégées en lecture ou écriture par différentes méthodes d'authentification.

Pour l'essentiel les architectures actuelles utilisant les cartes à puces considèrent cette dernière comme un élément passif (essentiellement un système de gestion de fichier). Un proxy localisé sur le terminal réalise une conversion de protocole (par exemple HTTP [r7] ou NFS [r4]), réalisant la traduction d'un accès fichier en une série de commandes (APDUs) cartes.

L'usage des puces dans un réseau est un sujet controversé, par exemple on peut lire dans le *white paper* de Sun relatif à l'architecture iPlanet (chapitre 9 - Authentication), "*smartcards if lost disable the user*" , "*smartcard can break*".

Notre objectif est d'utiliser une puce non pas comme un système de gestion de fichier, mais plutôt comme un processeur capable d'accéder de manière autonome à internet.

Cette architecture novatrice (une dizaine de brevets ont été déposés) ouvre de nouvelles perpectives quant à la mise en œuvre des puces dans les réseaux et architecture distribuées.

4. Les architectures ouvertes

Historiquement les puces se sont développées autour d'applications spécifiques (cartes bancaires par exemple). Les systèmes d'exploitation embarqués (que l'on nomme *masque*, car ils sont intégrés lors de la fabrication de la puce) étaient réservés à des usages particuliers. Aujourd'hui on définit par architecture ouverte des systèmes d'exploitation permettant l'exécution de programmes écrits dans divers langages (Java, Basic ...).
En 1997 est apparu la JavaCard [p1] qui intègre une machine virtuelle Java, c'est le premier pas vers des architectures multi applications écrites en Java. Ce langage permet de réaliser des services qui autrefois auraient impliqués la conception (coûteuse) d'une carte particulière. Les applets logés dans les cartes peuvent implémenter des fonctions telles que porte monnaie électronique, programme de fidélisation ....
En 1999 Microsoft [p2] a annoncé Smart Card for Windows (SCW), une carte à puce qui sera disponible avec son prochain système d'exploitation, courant 2000. La carte (qui se programme en basic) est organisée autour d'un système d'exploitation largement diffusé, qui permet à son porteur de la mettre en œuvre facilement, cette approche consiste à associer à chaque objet présent dans la carte une API de niveau Système d'Exploitation.

5. La carte internet

Une première génération de ***carte internet*** (@card) [r9,p3] est en cours d'évaluation et fait l'objet de recherches financées pour partie par plusieurs conventions Cifre. Ces cartes ont pour objectifs de s'intégrer à la communauté internet et d'améliorer la sécurité (authentification, intégrité, confidentialité) nécessaire aux nouveaux services fournis par le réseau. Une première génération de @card (baptisée *iSimplify!*) sera disponible courant 2000.
Une carte partage la pile TCP/IP du terminal auquel elle est connectée, elle hérite donc de sa configuration réseau (adresse IP, masque réseau, passerelle, serveur DNS, etc...). Elle est organisée autour d'un serveur web, le protocole http permet d'accéder via des url (http://127.0.0.1:8080/index.html) aux objets logés dans la carte, qui peuvent être de diverses natures, simple fichier ou algorithme cryptographique (clé symétrique, asymétrique, génération de certificats ...). Elle est également en mesure de se connecter à des serveurs externes, un objet identifié par une *url carte à puce* sera physiquement logé dans une carte, ou stocké sur un serveur distant, la puce assurant si nécessaire la sécurité requise pour son transfert (instanciation) ou son utilisation déportée (via des messages sécurisés). C'est la notion de ***fichier virtuel***.
Classiquement le porteur d'une carte est identifié par son code secret (pin code), cette authentification peut être suffisante pour autoriser l'accès à des objets personnels tels que bookmark ou procédure de connexion à un serveur internet (hotmail, altavista ....). Mais dans la plupart des cas le porteur n'est pas le propriétaire des objets logés dans la carte, une fois qu'il est authentifié, le prestataire du service vérifie que la puce abrite l'un de ses objets.

5.1 Architecture TCP/IP d'un terminal, notion de SAP

Dans l'architecture TCP/IP une application réseau (telle que un navigateur web par exemple) s'appuie sur un modèle qui comporte les 4 premières couches du modèle OSI Les couches 1 et 2 représentent par exemple la carte

(physique) ethernet ou un modem. La machine voit (utilise) une ressource réseau (carte ...) à travers une couche d'interface logicielle particulière encore nommée driver couches basses (ainsi NDIS est une interface couche basse développée par Microsoft). De fait, utiliser une carte réseau dans un système c'est mettre en œuvre un driver couches basses, en terminologie ISO cette interface peut être vue comme un SAP (Service Access Point) de niveau 2 (SAP2). De façon analogue une application utilise les couches TCP/IP au moyen d'une bibliothèque réseau fournie par le système hôte, une application réseau sait mettre en œuvre une telle bibliothèque à travers des SAP de niveaux 3 (SAP3) ou 4 (SAP4). Par exemple winsock.dll est la bibliothèque dynamique qui permet d'utiliser TCP/IP sur une machine windows.

5.2 Notion d'APDUs

Les couches OSI 5 (session) et 7 (application) ont été combinées en une seule entité définie par le standard ISO 7816-4 . Une application logée dans une carte communique au moyen d'APDUs qui sont transportées par la ligne série. Une APDU contient un message de commande ou un message de réponse qui est envoyé ou reçu vers/depuis le lecteur. Le dialogue se déroule selon un paradigme du type question/réponse. Le terminal émet une question (.command) à laquelle la carte répond (.response), de fait la norme 7616-4 fixe les règles du dialogue entre la carte et le terminal de manière analogue à une couche session. Une commande APDU comprend au minimum 4 octets nommés CLA INS P1 P2

5.3 Mise en œuvre des APDUs

Côté carte une entité parfois appelée APDU Manager analyse le flux des APDUs entrantes et réalisent les opérations nécessaires. Par exemple dans une carte multi applications, une APDU *SELECT* est utilisée pour sélectionner une application particulière (un programme écrit en java, ou encore un cardlet). Cette opération étant réalisée, toutes les APDUs reçues sont routées vers l'application en cours. Dans un terminal un logiciel particulier gère le lecteur de cartes. En 1996 un ensemble d'industriels de la carte à puce et de systèmes informatiques ont adopté le standard PC/SC qui permet l'intégration de lecteurs et de cartes dans les machines informatiques. Une application émet/reçoit des APDUs vers/depuis un lecteur à l'aide d'APIs (Application Programmatic Interface) délivrées par le système d'exploitation. Nous appellerons AMUX (APDU multiplexeur) la couche logicielle logée sur le terminal ou la carte qui réalise la commutation des APDUs vers les applications utilisatrices.

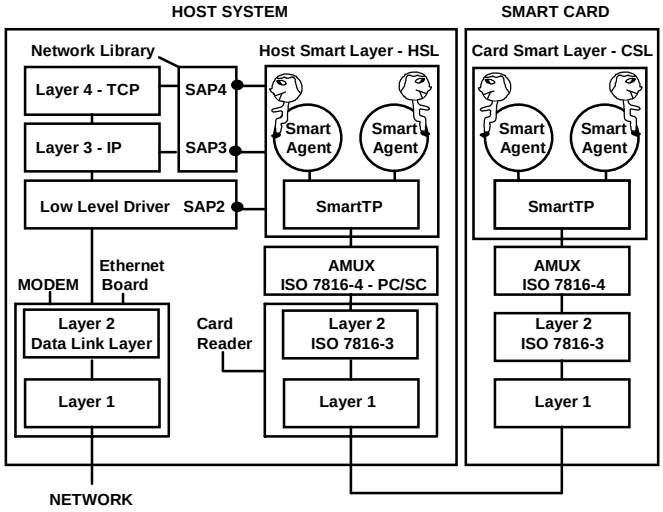

Figure 1 - Carte à puce Internet

5.4 Architecture d'un carte internet.

L'architecture (Cf figure 1) comporte deux piles protocolaires (Smart Layer) une logée sur le terminal (Host Smart Layer HSL) et l'autre dans la puce (Card Smart Layer - CSL).

Chaque pile s'appuie sur des couches OSI 1 et 2, définies en fait par les normes ISO 7816. Nous avons défini un protocole inspiré de TCP (Smart Transfer Protocol - SmartTP [r9]) qui est transporté par des APDUs spécifiques, c'est à dire qu'une entité SmartTP émet et reçoit des APDUs vers/depuis une couche AMUX.

Une unité de donnée SmartTP (SmartTP pdu, protocol data unit) comporte une référence de destination, une référence source, et un champ flag de 8 bits. Une référence est l'équivalent d'un port TCP ou UDP.

L'entité SmartTP se comporte comme un commutateur qui dirige les SmartTP pdu depuis/vers des entités logicielles autonomes nommées Agents.

Deux Agents sont mis en relation par une session, et échangent des SmartTP pdu. Un flag OPEN définit l'ouverture d'une session, un flag CLOSE indique la fermeture d'une session. Une agent peut se bloquer c'est à dire émettre un pdu avec un flag BLOCK positionné, et suspendre toute activité avant la réception d'un pdu. Deux flags additionnels (READ et WRITE) permettent de construire le paradigme OPEN-READ-WRITE-CLOSE (bloquant ou non) utilisé dans les systèmes Unix pour la manipulation des fichiers.

Côté terminal un agent réseau peut accéder à un SAP (par un exemple un SAP de niveau 4, c'est à un point d'accès sur la couche TCP ou UDP du terminal), il se comporte en fait comme un traducteur de protocole TCP/SmartTP ou UDP/SmartTP.

Une session entre un agent carte web (qui implémente le protocole http) et un agent terminal réseau (par exemple un serveur TCP sur le port 8080) permet de transformer une carte à puce en un serveur web qui utilise la pile TCP/IP du terminal auquel la puce est reliée mais qui réalise le protocole HTTP.

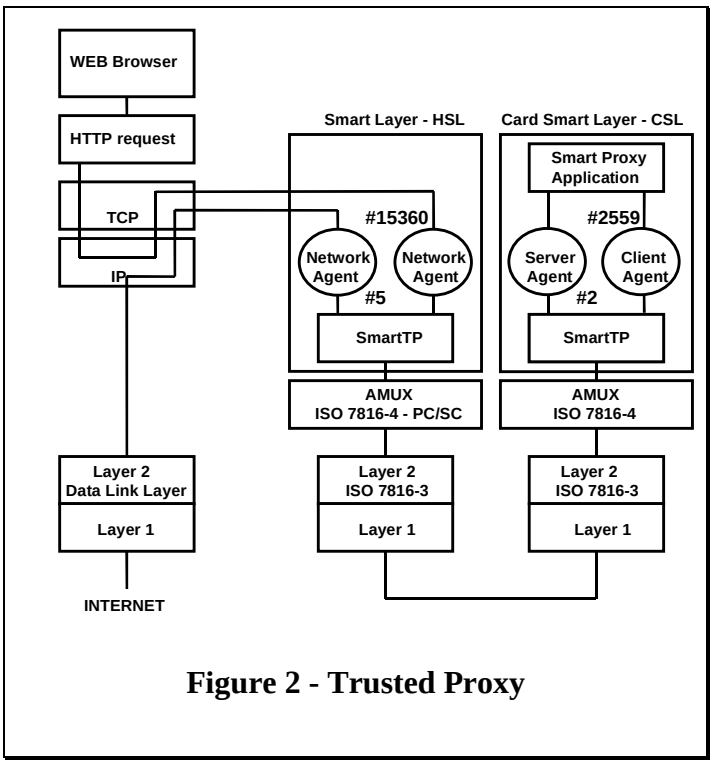

Figure 2 - Trusted Proxy

Une application carte mise en œuvre à partir d'un URL (c'est une notion similaire au CGI) peut accéder au web à l'aide d'une session agent carte/agent réseaux client TCP, l'association de deux sessions simultanées permet de réaliser un proxy qui relie par exemple un navigateur à un serveur externe. C'est la notion de *trusted proxy* (Cf figure 2)

5.5 Un exemple de session http

Le navigateur (T) ouvre une connexion TCP avec un agent réseau (référence 15360), une session est ouverte avec un agent carte (C) dont la référence est 2.

T[s=15360,d=2,open+block],

C[s=2,d=15360].

La requête http est transmise depuis l'agent réseau vers l'agent serveur web.

T[s=15360,d=2,write+block,data],

C[s=2,d=15361].

La dernière partie de la requête est transmise**,**

T[s=15360,d=2,write+block,data].

Le serveur carte transmet le fichier et son entête**.**

C[s=2,d=15360,write+block,data],

T[s=15360,d=2,block].

C[s=2,d=15360,write+block,data],
T[s=15360,d=2,block].
Le serveur carte met fin à la session**,**
C[s=2,d=15360,write+close,data].

6. Objets virtuels multimédia

Dans ce contexte nous avons défini la notion d'***objet virtuel*** éventuellement multimédia. L'objet virtuel est un objet numérique que l'on peut utiliser sans support physique au moyen du réseau. Un tel objet est associé à des *méthodes d'accès* plus ou moins sécurisées et soumis à des *contraintes de débit,* sa mise en œuvre peut nécessiter une *configuration* particulière du terminal.
On peut considérer que l'accès à une messagerie électronique est un objet virtuel, un logiciel capable de lire ou de visualiser le courrier est une méthode associée à cet objet, la sécurité est réalisée par un login et un mot de passe.
De même le contenu d'un album cédérom est un objet virtuel. La technologie de compression au format MP3 permet d'accéder à cet objet avec des débits de l'ordre de 0.5 à 1Mo/minute compatibles avec les capacités du réseau internet. Notre vision du lecteur cédérom est en fait un terminal banalisé connecté à internet. Les enregistrements sont logés sur un serveur. L'utilisateur acquiert un certificat de propriété (définitive ou temporaire) qui est stocké dans sa carte à puce, ou sur un serveur distant. L'objet qu'il possède (CD) n'a pas de support physique, il s'agit bien d'un objet virtuel. Un des avantages pour le propriétaire est la composition de programmes de son choix à partir d'un terminal quelconque et d'une carte à puce.
De fait de nombreux objets virtuels font leur apparition sur internet, radio internet, journaux électroniques, télévision web. La carte à puce internet a pour objectif de contrôler les accès à ces nouveaux services lorsqu'ils ne sont pas gratuits. Eventuellement cette opération constitue une méthode de rémunération en soit, par exemple la carte certifie que l'accès à un service est réalisée grâce à un *internet service provider* (ISP) particulier.

6.1 L'omniprésence d'internet

Une autre contrainte est de pouvoir accéder à un objet virtuel à partir de terminaux variés . Nous rentrons dans l'ère de l' omniprésence des ordinateurs (*ubiquitous computing* [e1]). Ce qui signifie que de plus en plus d'objets courants intègrent un microprocesseur, et ont la capacité de se connecter à internet (selon *Frost & Sullivan* 40 % des dispositifs connectés à internet en 2001 ne seront pas des ordinateurs personnels).
Les terminaux internet (multimédia par nature, puisque incorporant un navigateur) tendent à se banaliser. Un utilisateur (parfois mobile) utilisera plusieurs types de terminaux reliés à internet dont il ne sera pas forcément le propriétaire, par exemple

- Le téléphone mobile, le réseau GPRS permettra l'accès à internet depuis un mobile en France vers le début 2001, mais des solutions propriétaires existent déjà (Nokia...)
- Les consoles de jeux (Dreamcast de Sega ...)
- La télévision munie d'un set top box (Netgem, Netbox ...)
- Les organisateurs (Psion, Palm, Pilot VII..)
- Les mini portables

L'objet virtuel multimédia est accessible à partir d'une pluralité de terminaux qui devront si nécessaire être *configurés* de manière dynamique. Par exemple le terminal devra charger un applet spécifique pour visionner une image animée codée selon un format spécifique.
Dans notre approche la carte à puce est l'amorce indispensable pour l'instanciation d'un objet virtuel sur un terminal.

6.2 Objets virtuels multimédia distribués.

Ces objets sont accessibles grâce à un terminal banalisé connecté au réseau internet. La carte à puce internet présente à ce dernier, en fonction des ressources dont

il dispose un annuaire (*bookmark*) des services disponibles.
L'utilisateur sélectionne un objet virtuel particulier (lecture d'un journal, écoute d'un enregistrement sonore...)
La carte internet réalise alors la connexion au serveur requis, authentifie le client du service, configure si nécessaire le terminal et assure (pour tout ou partie en fonction du débit imposé par l'objet virtuel) la confidentialité des données échangées
Dans notre approche la carte internet est le titre de propriété des objets virtuels.

Figure 3 - Architecture de mise en œuvre des objets virtuels.

6.3 Architecture de mise en œuvre de l'objet virtuel.

L'architecture de mise en œuvre d'un objet virtuel est constitué d'un terminal connecté à internet, d'un carte à puce, et d'un ensemble de serveurs.

- Les droits d'une carte peuvent être logés dans une base de donnée embarquée ou sur un serveur externe (*User Data Base Server*).
- Un serveur d'objet virtuel (*Object Server*) délivre un contenu (journal, multimédia ...)
- Un logiciel particulier (par exemple un applet ou un plug in que nous nommerons *Viewer*) peut être nécessaire pour la mise en œuvre d'un objet virtuel sur le terminal, et est disponible sur un serveur particulier (*Viewer Server*). La licence d'un tel logiciel peut résulter d'un accord avec le fournisseur du contenu, ou obtenue sur la base d'un paiement par usage ("pay per use").
- Un serveur de paiement (*Payment Server*) peut être utilisé pour le paiement du service

7. Objets mobiles embarqués

Le commerce de l'objet virtuel pose le problème de l'identification des objets logés dans un puce, et de l'échange de messages entre une communauté de client et de serveurs. Une approche objet distribué peut être une réponse adaptée aux problèmes de normalisation et d'inter opérabilité.

7.1 Des objets serveurs embarqués et mobiles

Grâce au mécanisme de *fichier virtuel* un applet virtuellement stocké dans une carte à puce (l'adresse du serveur est 127.0.0.1, c'est à dire l'adresse de boucle du terminal) peut communiquer avec cette dernière en utilisant des protocoles tels que TCP ou UDP...

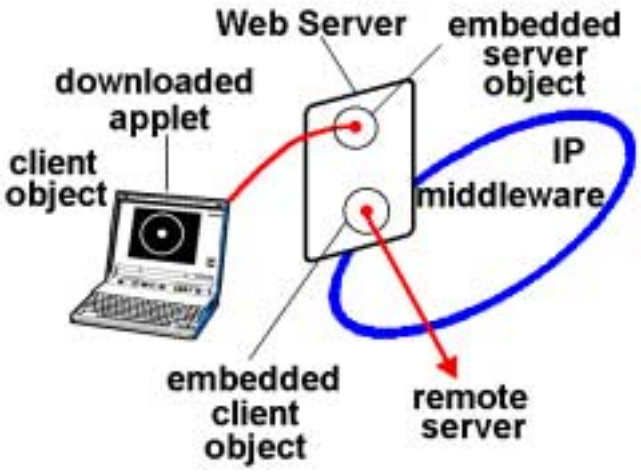

Figure 4 - Objets mobiles embarqués.

Il est donc possible d'accéder à des objets logés dans une carte à puce (par exemple un porte monnaie électronique) depuis un applet incrusté dans une page html. Différentes architecture sont envisageables citons pour mémoire les principales, *JavaRMI* qui s'appuie sur un

environnement Java, ou *DCOM* qui est basé sur le déport d'API et plus particulièrement destiné à un environnement windows. Dans ces deux exemples la carte joue le rôle du serveur web qui contient les objets clients et réalise toutes les opérations nécessaires pour l'échange des messages (par exemple dans le cas de *JavaRMI* la carte comporte le serveur d'enregistrement).

Un inconvénient des architectures les plus utilisées est d'être lié à des modèles particuliers (Java, win32 ...). Une carte à puce est un système embarqué, de surcroît mobile et à priori indépendant des plate-formes informatiques avec lesquelles elle communique. Cependant il convient de remarquer qu'il existe des Java Card mais également des Smart Card for Windows, on peut donc craindre que des objets (serveurs) transportés par des cartes soient dédiés à des architectures propriétaires (en particulier utilisant des navigateurs IE ou-exclusif Netscape).

Une autre question soulevée par un modèle d'objet (serveur) mobile est la compatibilité en terme de complexité des modèles existant avec les capacités mémoires relativement limitées des cartes à puces actuelles.

Une réponse élégante à cette problématique est la définition d'une architecture basée sur XML qui permettrait d'utiliser les objets serveurs cartes indépendamment de la plate-forme qui accueille les clients auxquels ils sont connectés. A titre d'exemple citons le protocole SOAP (Simple Object Access Protocol) qui permet l'échange de messages en syntaxe XML, et transportés par le protocole HTTP. Le langage XML permet d'une part de décrire les messages des requêtes de mais également d'exprimer les réponses (qui comporte éventuellement des objets multimédia) indépendamment du terminal utilisé (téléphone mobile, PC, kiosque internet..).

Remarquons enfin que l'objet serveur peur être virtuellement logé dans la carte à puce, c'est à dire que cette dernière se comporte comme un proxy d'accès (***trusted proxy***), qui fournit la sécurité nécessaire pour l'accès à des objets distants à travers le réseau internet.

7.2 Des objets clients embarqués et mobiles

En raison de son espace mémoire réduit, une puce peut utiliser des objets (serveurs) distribués à travers internet, des objets clients sont donc embarqués, le réseau internet est utilisé comme un bus de communication (middleware IP) entre la carte et des serveurs distants. Par un exemple un service qui n'est pas à priori logé dans la carte peut être disponible sur une base de donnée externe. Le service peut être utilisé à distance ou chargé dans la carte, et peut être vu dans ce cas comme un composant invoqué de manière dynamique.

- Une première contrainte est de disposer d'une architecture de communication compatible avec les contraintes introduites par les firewall ou les proxy http. Les messages doivent être transportés par les protocoles supportés par ces dispositifs de sécurité.

- Une deuxième contrainte est de disposer de protocoles compatibles avec les capacités mémoires et les puissances de traitement des cartes à puces.

7.3 Un rapide panorama des architectures possibles.

Nous présentons un bref panorama des architectures disponibles, en remarquant toutefois qu'un choix particulier peut être imposé par des contraintes technologiques particulière (Java ...). De manière générale une architecture d'objets distribués comporte une entité *serveur* qui réalise effectivement des opérations sur des objets, et une entité *cliente* qui comporte seulement une description des méthodes disponibles sur le côté serveur. Une *interface* est présente côté client ou serveur qui réalise la description des opérations possibles sur les objets. Cette interface est décrite dans un langage indépendant de la machine utilisé, nommé *Interface Definition Langage* (IDL). L'interface est intégrée au code du client et du serveur sous forme de *stub*. Le stub convertit l'invocation de fonctions en messages qui sont transportés sur le réseau par divers protocoles, tels RPC, IIOP, HTTP, SOAP...

Serveur/client
Interface
Transport

RPC

Un schéma simple peut utiliser l'appel de procédure éloigné (par exemple DCE/RPC), une interface est identifiée par un *Uuid* (Universal Unique Identifier).Deux scénarios sont envisageables appel local, ou depuis le réseau, ce qui peut impliquer l'enregistrement de l'objet serveur à un *service des noms*. Par exemple un objet porte monnaie électronique logé dans une carte à puce comporte deux méthodes, pme.add et pme.sub il est associé à une interface interface_pme :

```
{ int Add ( [in] int x, [out] int *solde)
  int Sub ( [in] int x, [out] int *solde) }
```

L'usage d'UDP est bien adapté à un lien carte/navigateur qui est sans erreurs, par contre une utilisation à travers le réseau sera plus efficace avec TCP.

CORBA

L'architecture CORBA est organisée autour d'un *Object Request Broker* (ORB ou agent) associé au client et au serveur. Cet agent est responsable de la localisation du serveur associé à une interface particulière (le stub serveur est nommé *skeleton*), et permet également l'accès à des *services objets.* Les messages échangés par les agents (sept en tout) sont décrits par le protocole *General Inter ORB Protocol* (GIOP). Le transport des messages est assuré par le protocole *Internet Inter-ORB Protocol* (IIOP) qui s'appuie sur TCP. En complément à GIOP le protocole ESIOP (*Environnement Specific Inter-ORB protocol*) permet d'utiliser une pile de transport autre que IIOP (par exemple DCE/RPC). Les messages échangés entre client et serveur comportent l'identifiant d'une interface, la méthode invoquée, et la référence d'un objet sur laquelle s'applique l'opération. Lorsque le client ne comporte pas le code du stub mais obtient l'interface sur un serveur externe on parle *d'invocation dynamique* d'un objet (par opposition à l'*invocation statique*).

Client/serveur
Stub/Skeleton
ORB
GIOP/IIOP
TCP/IP

Une adaptation des cartes à CORBA a été proposée par [o1], au moyen d'un proxy ORB nommé *Card Object Adapter* (COA) qui traduit les messages ORB sous formes d'APDUs. Dans notre approche la carte traite directement les requêtes RPC ou le protocole IIOP.

Java RMI

RMI Registry	Registry
Client (.class)	Serveur (.class)
Stub (.class)	Skeleton (.class)
Remote Reference Layer	Remote Reference Layer
Transport	Transport

La technologie Java RMI (*Remote Method Invocation*) est une technologie propriétaire (Sun) destinée à l'usage d'objets distribués. Chaque entité client ou serveur comporte trois couches, la couche *stub/skeleton* qui comporte le code byte (.class) des interfaces clients ou serveur, la couche *Remote Reference Layer* en charge de la fonction de *marshaling* (encodage des messages) et la couche transport responsable du transport des messages (à l'aide du protocole propriétaire JRMP). Un des points forts du protocole de transport est d'utiliser http (*RMI Wire Protocol*) lorsque un firewall ou un proxy rend impossible la mise en œuvre de TCP. Une des originalités de Java RMI par rapport à CORBA est d'autoriser le transfert d'objets java dans les paramètres d'appel de méthode (à l'aide d'un mécanisme nommé *Object Serialization*). Le sub et le skeleton sont générés à l'aide du compilateur dédié *rmic*. Une JavaCard utilisant un sous ensemble restreint du langage Java une adaptation est nécessaire pour la génération d'un code byte compatible avec la machine virtuel de cet environnement.

L'objet serveur est enregistré sur une entité nommé *Registry* (serveur de noms), à laquelle il fournit également le code byte du stub (Registry.bind). Le code du client contient l'appel au Registry (Registry.lookup) grâce auquel il obtient en particulier le stub. Lorsque le client est inclus dans un applet, le serveur et l'entité Registry sont logés sur le même système hôte.

- Un serveur carte doit donc incorporer l'entité Registry.
- Un client devrait charger dynamiquement un stub, ce qui peut poser des problèmes de faisabilité et sécurité.

Java RMI s'est ouvert vers la norme CORBA (RMI-IIOP) afin de permettre une plus grande inter-opérabilité. Le compilateur rmic a été modifié pour produire des interfaces (stub & tie skleleton) qui utilisent un *sous ensemble* du langage IDL, le protocole IIOP est utilisé pour le transport des messages (et en particulier pour la sérialisation des objets, à la place du protocole JRMP). *Il n'est cependant pas possible de produire du JavA RMI à partir de l'IDL standard*.

DCOM

Client	Objet Serveur
Interface Proxy	Interface Objet
Proxy	Stub
APIs	APIs
SCM	SCM
RPC	RPC

Le modèle objet DCOM (*Distributed Component Object Model*) de Microsoft définit la notion d'*interface* (un tableau de pointeur sur des fonctions en fait) identifiée par un GUID (Globally Unique Identifier), et la notion d'*objet* qui contient des données et le code des fonctions. L'architecture COM est basé sur un jeu restreint d'APIs. Un service SCM (*Service Control Manager*) détermine à partir d'un identifiant d'une classe, le nom et la localisation du serveur qui gère cette dernière. Le transport des messages est assuré par un mécanisme du type DCE/RPC. Un client est associé à proxy qui comporte une interface et déporte tous les appels (via RPC) vers le stub serveur. Le stub serveur comporte une interface vers un objet crée par le serveur. La technologie DCOM est propriétaire, elle sera probablement disponible pour les Smart Card for Windows (SCW).

SOAP

Le protocole SOAP (Simple Object Access Protocol) est une alternative à RPC proposée par Microsoft. SOAP utilise XML pour décrire les messages échangés entre objets et HTTP (sécurisé éventuellement par SSL) pour leur transport. Par exemple un objet dont l'interface est décrite par une DTD PME-URI est associé à une méthode PmeAdd. Le paramètre Somme est identifié par l'élément <Somme>. Les éléments Enveloppe et Body sont définis par une DTD propre au protocole SOAP ("urn:schemas-xmlsoap-org:soap.v1").

```
POST /PME HTTP/1.1
SOAPMethodName: PME-URI#PmeAdd

<SOAP:Enveloppe
xmlns:SOAP="urn:schemas-
xmlsoaporg:soap.v1"
<SOAP:Body>
<m:PmeAdd xmlns:"PME-URI">
<Somme>1000$</Somme>
</m:PmeAdd>
</SOAP:Body>
</ SOAP:Envelope>
```

```
HTTP/1.1 200 OK

<SOAP:Enveloppe
xmlns:SOAP="urn:schemas-xmlsoap-
org:soap.v1"
<SOAP:Body>
<m:PmeAddResponse xmlns:"PME-URI">
<Solde>1500$</Solde>
</m:PmeAddResponse>
</SOAP:Body>
</ SOAP:Envelope>
```

La réponse identifie la méthode invoquée (PmeAddResponse) et comporte l'élément <Solde>, qui est en fait la valeur retournée après l'exécution de la méthode PmeAdd.

8. Conclusion

Nous avons introduit un concept novateur de puce internet, destinés à améliorer la sécurité des transactions à travers la réseau, et à faciliter le commerce des objets virtuels. Une approche *objet embarqué* semble une voie prometteuse pour la normalisation et l'utilisation des ressources gérées par une carte à puces.

9. Références

9.1 Normes cartes à puce

[n1] ISO 7816-1 - Cartes à circuit(s) intégré(s) à contacts
Partie 1: Caractéristiques physiques - 1987

[n2] ISO 7816-2 - Cartes à circuit(s) intégré(s) à contacts
Partie 2: Dimensions et emplacements des contacts - 1988

[n3] ISO 7816-3 - Cartes à circuit(s) intégré(s) à contacts
Partie 3: Signaux électroniques et protocoles de transmission - 1989

[n4] ISO 7816-4 - Identification Cards - Integrated circuit(s) cards with contacts
Part 4: Inter-Industry commands for interchanges - 1995

[n5] ISO 14443-Identification cards - Contactless integrated circuit(s) cards - Proximity Cards
Part 1: Physical characteristics

[n6] ISO 14443-Identification cards - Contactless integrated circuit(s) cards - Proximity Cards
Part 2: Radio Frequency Power and signal interface

[n7] ISO 14443-Identification cards - Contactless integrated circuit(s) cards - Proximity Cards
Part 3: Initialization and Anticollision

[n8] Specification of the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface - GSM 11.11 - ETSI - 1997

[n9] Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface - GSM 11.14 - ETSI - 1997

9.2 Architecture

[a1] F.Guez C.Robbert A.Lauret - "Les cartes à microcircuit" - MASSON 1988

[a2] Le musée de la carte http://www.cardshow.com/museum/exposition.html

[a3] R.Merckling - A.Anderson - "Smart Card Introduction" - Request For Comment 57 (rfc 57) - 1994

[a4] Carol Hovenga Fancher, "In Your Pocket: Smartcards", IEEE Spectrum, pp. 47 - 53, February, 1997.

[a5] Roger Dettmer, "Getting Smarter", IEEE Review, pp. 123 - 126, May 1998

[a6] Jim Walejko "SmartCard Architecture" - NTU CA714-CA Graduate Computer Architecture http://www.cs.berkely.edu/~neefe/ntu.fa98/jim.project.html - 1998

[a7] Siu-Cheung Charles «An Overview of Smart Card Security» http://home.hkstar.com/~alanchan/papers/smartCardSecurity - 1997

[a8] Jean Pierre Tual "MASSC, A Generic Architecture for Multiapplication Smart Cards", IEEE Micro September-October 1999.

9.3 Commerce électronique

[e1] Group D, I.S.206, SIMS - "Future Organisation of the Computer and Communications Industries" - 1997 http://www.sims.berkeley.edu/courses/is206/f97/GroupD/dfutre.html

[e2] Constantinos Markantonakis, "The Case for a Secure Multi-Application Smart Card Operating System", Information Security Workshop 97 (ISW'97), September 1997 (Ishikawa in Japan), Springer-Verlag (LNCS 1396), Berlin (1997), pp.188-197

[e3] Robert W.Baldwin & C.Victor Chang - "Locking the e-safe" - IEEE Spectrum pp 40-46 - February 1997

[e4] Michael C.McChesney - "Banking in cyberspace: an invesment in itself" - IEEE Spectrum - pp 54 - 59 - FEBRUARY 1997.

9.4 Réseau

[r1] Ton Verschuren - "Smart access: strong authentification on the Web" - Computer Network ans ISDN Systems n°30 - pp 1511-1519 - 1998

[r2] ISI - IBM Smart Card Identification Protocol http://www.iscit.surfet.nl

[r3] ACTIV CARD «ActivCard Synchronous Authentification - A white paper» http://www.activecard.com/pressrom/whitepapers/des.html

[r4] Naomaru Itoi - Peter Honeyman - Jim Rees - «SCFS: A UNIX Filesystem for Smartcards» CITI Technical report 98-8 - December 1998

[r5] Naomaru Itoi - Peter HoneyMan «SmartCard Integration with Kerberos V5» CITI Technical Report 98-7 - December 1998

[r6] Matt Blaze - «High Bandwidth Encryption with Low-Bandwidth Smartcards»-

ftp://ftp.research.att.com/dist/mab/cad_cipher.ps - December 1995

[r7] Jon Barber "The Smart Card URL programming Interface" GDC'99

[r8] Pascal Urien & Hayder Saleh - «Une nouvelle approche de la carte à puce réseau» - Communication JRES 99 - Montpellier - 29-11/03-12 99

[r9] Pascal Urien, Hayder Saleh, Adel Tizraoui "La puce Internet, une architecture ouverte adaptée aux applications distribuées multimédia sécurisées". Infosec'Com 2000 - 7,8 juin 2000 La défense.

9.5 Sécurité

[s1] Bruce Schneier - Adam Shostack «Breaking Up Is Hard To Do: Modeling Security Threats for Smart Cards» - Usenix 99

9.6 Objets et cartes à puces

[o1] Patrick Biget, Patrick Geaorge, Jean Jacques Vandewalle "How smart cards can benefi from object-oriented technologies" Future Generation Computer Systems 13 (1997) pp 75 -90

9.7 Divers

[p1] Shlumberger "Using a high level programming langage with a microcontroler" Patent WO 98/19237

[p2] Microsoft «SmartCard White paper» http://www.microsoft.com/windwsce/smartcard/ressources/wp2.asp - 1999

[p3] Pascal Urien - "Procédé de communication entre une station d’utilisateur et un réseau, notamment de type internet, et architecture de mise en œuvre" brevet déposé le 13 août 1988, N° d’enregistrement 98 10401, N° de publication 2 782 435