$N°$d'ordre : 2969

THÈSE

présentée à

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE

par Marcien Mackaya

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Test des protocoles et services liés à la mobilité

Soutenue le 11 Mars 2005 après avis de :

A. CAVALLI Rapporteurs
C. VIHO

Devant la commission d'examen formée de :

S. CHAUMETTE	Professeur	Président
A. CAVALLI	Professeur	Rapporteurs
C. VIHO	MCF HDR	
R. CASTANET	Professeur	Examinateurs
P. COMBES	Ingénieur Expert	

Remerciements

Puisqu'une thèse, malgré tout, n'est pas le fruit d'une oeuvre solitaire, je tiens à remercier :

- Monsieur Richard Castanet, Professeur des Universités, pour m'avoir encouragé à entreprendre ce travail, pour m'avoir encadré tout au long de son déroulement, et pour avoir toujours mis à ma disposition toutes les ressources nécessaires à mon épanouissement.

- Madame Ana Cavalli, Professeur à l'Institut National des Télécommunications à Evry et Monsieur César Viho, Maître de Conférences HDR à l'Université de Rennes 1, pour avoir accepté de rapporter ce travail.

- Monsieur Serge Chaumette, Professeur à l'Université de Bordeaux 1 qui m'a fait l'honneur de présider mon jury.

- Monsieur Pierre Combes, ingénieur expert à France Télécom RD, d'avoir accepté de faire partie de mon jury.

- Sébastien Grivet et Ousmane Koné pour leurs conseils et leur aide.

J'ai gardé d'excellents souvenirs de ma collaboration avec Fathia Zaïdi, Cedric Besse, Wafa Mederreg et tous les membres du projet PLATONIS.

J'ai également une pensée amicale pour l'ensemble des collègues avec lesquels j'ai partagé ces années de thèse : je pense en particulier à Laurent Tichit, Maître de Conférences à l'Université de Marseille II mais aussi à Bertrand Cirou, Olivier Parisy, Anthony Don, Benjamin Taton, David Renault, Bertrand Kerautret, Yon Dourisboure, Afif Sellami, Pierre Hanna, Sylvie Alayrangues, Jean-Baptiste Leproux, Pascal Grange et bien d'autres encore !
Je pense aussi à mes compatriotes grâce auxquels je n'étais plus "seul" au LaBRI, il s'agit de Rodrigue et de Joan qui est nouvellement arrivé.

A J. Tati, B. Larissa, N. Clara, R. Safou, M. Blanchard, B. Arsène, ma famille bordelaise, merci d'avoir toujours été là.

Messieurs Pierre Moukéli, Louis James Ndoutoume et J.C. Hochon, vous avez été déterminants pour que je sois ici aujourd'hui, je vous en remercie du fond du coeur.

Je remercie "ya" Jean Marie, un grand frère exemplaire qui a toujours su tenir ce rôle, et Léandre, autre grand frère sans qui cette aventure n'aurait sans doute pas pu avoir lieu, un soutien de tous les instants.
Enfin, à mes parents puis à l'ensemble de ma famille, je tiens à vous exprimer ici toute ma gratitude.

Test des protocoles et services liés à la mobilité

Résumé : Les fortes avancées technologiques dans le domaine des réseaux entraînent la conception de nouveaux protocoles et la création de nouveaux services. Ces nouveaux protocoles et services mettent en relation des éléments hétérogènes qu'il convient de tester pour garantir leur conformité et leur interopérabilité. Pour ce faire, nous avons adopté les normes WAP et UMTS et proposé une méthodologie de test de la couche applicative WAE (à travers une application écrite en WML), des couches protocolaires WSP et WTP du WAP et des services de localisation dans le réseau UMTS. Ces tests ont été par la suite mis en ouvre dans une plate-forme afin de vérifier que les échanges protocolaires sont conformes aux spécifications et que les différentes entités peuvent interagir correctement.

Mots-clé : Test de conformité, test d'interopérabilité, WAP, GSM, GPRS, UMTS, WAE, WML, WSP, WTP, plate-forme.

Testing protocols and services based on mobility

Abstract : The strong advances in the field of networks involves the design of new protocols and the creation of new services. These new protocols and services connect heterogeneous elements which must be tested to guarantee their conformity and their interoperability. With this intention, we adopted the WAP and UMTS standards, and proposed a methodology to test the WAE applicative layer (through an application written in WML), WAP WSP and WAP WTP layers, and UMTS location services. Thereafter these tests have been implemented in a real platform in order to check that the exchanges are in conformity with the specifications and that the various entities can interact correctly.

Keywords : Conformance test, interoperability test, WAP, GSM, GPRS, UMTS, WAE, WML, WSP, WTP, platform.

Table des matières

Table des figures

Chapitre 1

Introduction

1.1 Motivations et cadre du sujet

1.1.1 L'Internet Mobile

Au premier trimestre 2004, l'Autorité de régulation des télécommunications (ART) a annoncé que 41,9 millions de Français disposaient d'un téléphone mobile, ce qui représente un taux de pénétration de 69,5%.
Parallèlement, le nombre d'utilisateurs d'Internet ne cesse de progresser, d'après l'Observatoire des Usages Internet (OUI), la population internaute a franchi pour la première fois au premier trimestre 2004 le seuil des 23 millions d'individus (plus exactement, en mars 2004, 23 089 000 Français âgés de 11 ans et plus se sont connectés à Internet au cours du dernier mois), soit 45% de la population âgée de 11 ans et plus. Le nombre d'internautes affiche ainsi une progression de 14% en un an.
Dans un tel contexte, quoi de plus naturel que de penser faire de l'Internet avec son terminal mobile, fruit de la rencontre de deux des domaines les plus dynamiques à l'heure actuelle. Toutes les technologies et les services de l'Internet Mobile sont promis à un bel avenir, d'autant que ces tendances ne sont pas propres au marché français mais qu'elles s'observent dans tous les pays développés.

1.1.2 Les enjeux

A l'heure actuelle, le principal enjeu de l'Internet Mobile est l'adaptation du monde Internet, tel qu'on le connaît aujourd'hui, aux téléphones mobiles. Cet objectif passe par une *redéfinition des services*, l'amélioration de l'ergonomie et une *meilleure interopérabilité* des terminaux avec les passerelles et les serveurs Internet.
Ensuite, le deuxième enjeu sera d'ouvrir Internet à l'ensemble des abonnés au téléphone mobile, de proposer des *services innovants* et d'inciter ces abonnés à adopter de nouveaux usages.

Pour répondre à ces défis, les solutions devront proposer :
- une facilité d'utilisation,
- des terminaux à un prix accessible,
- des protocoles et services adaptés à la mobilité

Plusieurs technologies ont été élaborées pour permettre d'accéder aux services temps réels, personnalisés et adaptés à la mobilité : le WAP[1] sur GSM[2]/GPRS[3], l'i-Mode[4], l'UMTS[5], etc..

1.1.3 Les différentes technologies

Le WAP est une norme de communication permettant à des terminaux mobiles, téléphones portables ou assistants personnels numériques - PDA - comme les Palm Pilots ou les terminaux sous Windows CE 3.x de se connecter aux services par l'intermédiaire du réseau Internet.
Le WAP est normalisé par l'organisation WAP Forum [28, 29] qui a vu le jour en 1997, d'une alliance entre les principaux acteurs du domaine des télécommunications sans fil (Nokia, Ericsson, Motorola, Phone.com et Microsoft), il regroupe aujourd'hui plusieurs centaines de membres.
Le protocole WAP standardise donc l'échange d'information entre le terminal mobile et une passerelle qui permet la conversion des protocoles de transport de données entre Internet et le réseau mobile (voir figure 1.1)

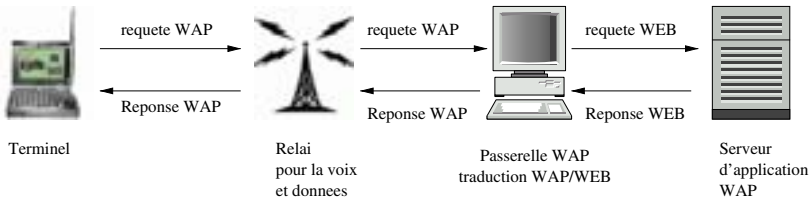


FIG. 1.1 – Architecture du WAP

L'une des forces du WAP est très certainement la séparation des services du type des réseaux de télécommunication utilisé. Cette séparation va pouvoir assurer la pérennité du WAP :
- Le réseau GSM : à l'heure actuelle la majorité des services WAP s'appuient sur les réseaux GSM existants. Ces réseaux sont peu adaptés à la transmission des données. Le débit qu'ils offrent (9600 bit/s) reste insuffisant pour couvrir les nouveaux besoins de transfert de données (e-mail, fichiers, pages Internet, etc..). Mais des nouvelles normes sont

[1] Wireless Application Protocol
[2] Global System for Mobile communications
[3] Global Packet Radio Service
[4] Service d'accès sans fil à l'Internet japonais
[5] Universal Mobile Telecommunication Services

en cours de déploiement et de développement. Elles devraient être à même de résoudre ces différents problèmes et de doper le marché du WAP. Ces normes sont : le GPRS et l'UMTS.
- Le réseau GPRS : c'est une extension de la norme GSM permettant le transfert de données en mode paquets. L'intérêt du GPRS pour le WAP est double. D'une part le débit de transport est supérieur à celui du GSM (114 kbit/s) et d'autre part, l'accès aux services WAP ne sera plus facturé à la durée mais au volume de données échangées.
- Le réseau UMTS : l'UMTS fait partie de la troisième génération de réseau mobile. Cette norme permettra d'offrir des débits allant de 64 kbit/s à 384 kbit/s, puis à terme 2 Mbit/s.

1.2 Le travail réalisé

Faire de l'Internet avec son terminal mobile aujourd'hui nécessite donc la conception de nouveaux protocoles et de nouveaux services. Comme on le voit sur la figure 1.1, ces services et protocoles mettent en relation des entités hétérogènes, ce qui pose alors le problème de la garantie de leur interfonctionnement. Ce travail de recherche contribue à l'Internet du futur et se place dans un contexte "qui doit apporter une réponse à l'hétérogénéité des besoins en communication" (rapport RNRT[6]-Internet du futur). Pour ce faire, nous avons d'une part adopté la norme WAP et d'autre part nous nous sommes tout particulièrement intéressés aux défis suivants, posés dans la section 1.1.2 :
- garantir l'interopérabilité entre les terminaux, les passerelles et les serveurs Internet
- garantir des applications et/ou services innovants : ce sont les services et/ou applications capables de susciter l'intérêt des abonnés pour l'Internet Mobile.

1.2.1 Garantie de l'interopérabilité et des services innovants

L'ultime recours pour s'assurer qu'un produit fait bien ce que l'on attend de lui est de le tester. Le test de ces nouveaux produits deviennent ainsi des activités stratégiques dans l'industrie du logiciel télécom, non seulement pour les opérateurs (France Télécom, Bouygues, Cegetel, LDCom, etc.) mais également pour les équipementiers (Alcatel, Siemens, Ericsson) et les outilleurs (Tekelec, etc.).

Dans le cadre de notre travail, les tests effectués sont basés sur l'utilisation de méthodes formelles et l'expérimentation. Les normes WAP étant

[6]Réseau National de Recherche en Télécommunications

décrites en langage informel (naturel), cela manque de précision, ce qui peut entraîner des problèmes dans la conception et la compréhension du protocole. Nous avons utilisé le langage SDL (Specification and Description Language) pour décrire les comportement des services et protocoles à tester, ce qui a comme avantage de permettre l'automatisation de la procédure de génération de tests.
L'expérimentation consiste à observer l'exécution des protocoles et services dans un environnement réel (sur une plate-forme réelle dans notre cas), tout en permettant le contrôle et l'observation de l'ensemble des comportements possibles contenus dans la spécification.
C'est donc grâce à la validation et à l'expérimentation sur des plates-formes réelles qu'on pourra garantir des services fiables.

Les services testés

Les services testés sont de trois ordres :

- services applicatifs : ces services correspondent à l'exécution de programmes interagissant directement ou indirectement avec les utilisateurs finaux munis de leur mobile. En effet, dans l'environnement WAP, la couche WAE fournit un cadre de spécification d'applications qui sont définies en langage WML[7]. Il a été possible de définir des enchaînements de traitements et de requêtes correspondant à des modèles généraux d'applications. Ces enchaînements sont la base pour la génération de séquences de test.
- services fournis par les couches protocolaires : ces services sont spécifés dans les normes. Nous nous sommes intéressés aux couches protocolaires WSP[8] et WTP[9].
- service innovant : un des challenges de l'Internet Mobile est de proposer des services innovants afin d'inciter les abonnés à l'adopter. A cet effet, les services basés sur la localisation ou géo-dépendant se positionnent comme une valeur de différentiation pour les opérateurs. Ce service met en oeuvre des mécanismes qui permettent au serveur de localisation (LCS-serveur) de calculer les positions (information de localisation ou LCS-information) des mobiles afin de les fournir au serveur WAP (LCS-client) sur demande autorisée. Le serveur WAP pourra ainsi renvoyer au terminal WAP des informations personnalisées qui tiennent compte de sa position. Le problème qui se pose ici est donc celui de fournir avec fiabilité la position du terminal.

[7]Wireless Markup Language
[8]Wireless Session Protocol
[9]Wireless Transport Protocol

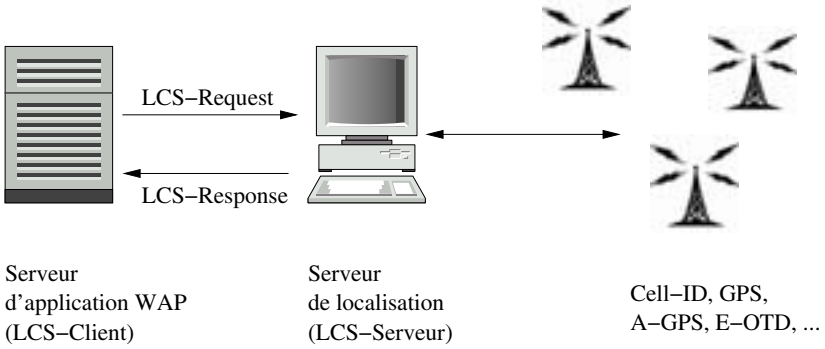


FIG. 1.2 – Architecture des LCS Services

Les tests réalisés

Les tests mis en oeuvre dans les architectures de test proposées sont les suivants :

- test d'application : ces tests doivent permettre de vérifier qu'une application est bien écrite et répond aux attentes des utilisateurs finaux. Nous avons donc proposé une méthode pour tester une application écrite en WML.
- test de services et de protocoles : conformité et interopérabilité locale "verticale", interopérabilité distante "horizontale". Pour ce faire, nous avons modélisé en SDL les protocoles à partir des spécifications (en langage naturelle le plus souvent) du WAP et effectué des simulations puis déduit des séquences de tests qui seront appliquées dans l'implantation réelle.
- test d'interfonctionnement :
  - ces tests doivent permettre de vérifier l'interfonctionnement entre une application sur un terminal et une application sur un serveur. Cette vérification implique l'observation des requêtes HTTP du serveur provoquées par celles du terminal. Il s'agit donc d'un test d'interopérabilité des services au sein de l'architecture de communication globale. Pour ce faire, nous avons proposé une méthodologie de vérification de l'intéropérabilité par la génération des fichiers de traces observés sur les interfaces installées au niveau de la passerelle, par le contrôle des requêtes émises à partir d'un PDA et par l'observation du réseau grâce au snifer ethereal.
  - pour fournir au LCS-client le LCS-infomation, plusieurs entités hétérogènes du LCS-serveur interopèrent, le test d'interfonctionnement va donc permettre de vérifier la fiabilité de la fourniture du LCS-information. Plusieurs travaux [35, 64] ont été publiés sur l'analyse de performance concernant les services de localisation, mais nous n'avons pas connaissance des études concernant la simulation fonc-

tionnelle et la conception de tests basés sur des techniques formelles dans les réseaux UMTS notamment. Pour ce faire, nous utilisons comme ci-dessus la technique de modélisation et de génération des tests qui garantissent la fourniture de l'information de localisation.

1.2.2 Mise en oeuvre des tests

Les tests sont donc mis en oeuvre dans une plate-forme afin de vérifier que les échanges protocolaires se passent bien et que les différentes entités peuvent interagir correctement. La plate-forme contient les éléments suivants :

- un terminal (simulateur, PDA, téléphone mobile, etc..)
- une passerelle en "open-source"
- un serveur d'application
- interfaces d'accès à la passerelle
- outils d'observation et de contrôle

L'originalité de notre plate-forme réside dans le fait qu'il s'agit d'une plate-forme de validation et d'expérimentation de l'interfonctionnement des protocoles et services. Et à notre connaissance, des plate-formes existent en France (VTHD : plate-forme d'interconnexion des réseaux à très haut débit, Amarrage : plate-forme multimédia basée sur des réseaux actifs, Parol : plate-forme basée sur l'ORB Jonathan, @IRS : Architecture Intégrée de Réseaux et Services) et aussi dans le monde (à savoir les plate-formes de l'ETSI, de Corée Telecom et de l'Université de Tsinghua en Chine), mais ces plates-formes ne sont pas destinées à la validation de l'interfonctionnement globale des protocoles et des services.

PLAN

Le *deuxième chapitre* présente le WAP et les technologies (GSM, GPRS, UMTS) qui le supportent. La deuxième section de ce chapitre présente le réseau GSM du point de vue de l'architecture et des services offerts. La troisième section décrit la nome WAP, son histoire, sa spécification, son architecture et sa pile protocolaire. Cette section relève les limites du WAP sur GSM et évoque la nécessité du GPRS (on parle souvent de GSM phase 2). La quatrième section de ce chapitre présente l'architecture du GPRS et les services offerts. Nous faisons état dans la cinquième section des services et architectures relatifs au réseau de 3^{eme} génération qu'est UMTS. La sixième section de ce chapitre va s'intéresser aux services de localisation dans le réseau UMTS car ce service sera modélisé dans le chapitre 4 afin de rendre fiable l'obtention de l'information de localisation. Cette section présente entre autre les différents types de services de localisation,

les différentes méthodes de localisation, l'architecture de ces services et un exemple d'application orientée localisation développée dans le cadre de ce travail.

Le *troisième chapitre* dresse l'état de l'art du test de conformité ou test dit "boîte noire". Nous présentons tout d'abord les différents types de test, nous évoquons la nécessité de développer des méthodes formelles de test et la classification de celles-ci tout en indiquant comment elles mettent en oeuvre des concepts essentiels (notamment la modélisation des entités, l'expression de la conformité et la technique de génération de test) formalisés par la norme Z500 [88]. La fin de ce chapitre présente ce qu'est le test d'interopérabilité et son contexte de recherche. dans la section 3.3 de ce chapitre, nous abordons le test d'interopérabilité, sa nécessité, sa problématique et le contexte de recherche. Enfin, dans la section 3.4, nous proposons une stratégie en vue de tester la conformité des composants et l'interopérabilité de notre système global.

Dans le *quatrième chapitre* présente une méthode de génération de test pour une application WML (ce qui va représenter le test de la couche applicative de notre pile WAP), cette méthode permet de vérifier qu'une application WML est bien écrite et qu'elle fournit les résultats attendus. Par la suite nous présentons la modélisation en SDL et la génération de test des protocoles WSP et WTP d'une part et du service de localisation (LCS-services, cas du réseau UMTS) d'autre part.

Le *cinquième chapitre* présente une méthode de validation de l'interopérabilité par contrôle et observation des fichiers de traces. Des résultats expérimentaux sont présentés en vue de valider l'interfonctionnement entre une application sur un terminal et une application sur un serveur grâce aux observations effectuées sur divers points stratégiques de la plate-forme. Enfin, le *sixième chapitre* conclut ce travail de recherche.

Pendant notre thèse, nous avons participé au projet PLATONIS[10] [52] labellisé par le RNRT et collaboré avec les chercheurs du SAMOVAR[11] (INT, Evry), Limos (Univ. Blaise Pascal, Clermont Ferrand), et France Télécom R&D. Cette collaboration a permis entre autre la mise en place de la plate-forme avec des sites chez chacun des partenaires. Le projet PLATONIS a fait l'objet des publications suivantes [14, 58, 16, 57, 15, 59].

[10] PLAte-forme de validaTion multI-protocoles et multi-Services
[11] Services répartis, Architecture, MOdélisation, Validation, Administration des Réseaux

Chapitre 2

Les réseaux mobiles : du WAP à l'UMTS

2.1 Introduction

Les deux avancées technologiques en télécommunication qui auront marqué le tournant du millénaire sont incontestablement l'accélération du développement d'Internet et la généralisation de la téléphonie mobile. Avec Internet il est possible d'avoir un accès instantané à une quantité infinie d'informations à partir de n'importe quel ordinateur relié à une simple ligne téléphonique ; la téléphonie mobile a quant à elle, permis d'affranchir l'utilisateur des contraintes géographiques. Il a donc été dans l'ordre des choses de penser faire de l'Internet avec son mobile.

Plusieurs normes ont vu le jour, mais la plupart des principaux acteurs du domaine des télécommunications (Nokia, Ericsson, Motorolla, Microsoft, etc..) ont mis en place le WAP forum afin de normaliser le protocole WAP qui apparaît aujourd'hui comme l'une des principales normes dédiée à l'Internet mobile.

L'une des grandes forces du protocole WAP est de pouvoir s'adapter à différents réseaux de communication mobile tels que GSM et GPRS. A l'heure actuelle, en France notamment, les services WAP s'appuient sur les réseaux GSM existants. Ces réseaux sont peu adaptés à la transmission des données, les débits qu'ils offrent (9600 bit/s) restent insuffisants pour couvrir les nouveaux besoins de transmission (e-mail, fichier, etc..). Ils sont également pénalisés par les temps de connexion et de réponse bien souvent trop longs. Mais n'accablons pas la norme WAP sur GSM car elle n'est que la première étape de "l'Internet Mobile" qui s'améliore avec l'arrivée progressive des nouveaux réseaux, en particulier le GPRS qui en constitue la deuxième génération. Le WAP sur GSM ou GPRS constitue un éclaireur pour "l'Internet Mobile".

Le GPRS sera malgré tout limité, notamment pour la fourniture des services multimédias, c'est ainsi que nombre de partenaires tels que l'ITU (International Telecommunication Union) et le 3GPP (3^{eme} Generation Partnership Project) s'attelle à la définition de la future technologie. Plusieurs normes de 3^{eme} génération existent, la principale étant l'UMTS (Universal Mobile Telecommunication System).

Par ailleurs, l'un des enjeux de l'Internet mobile sera d'ouvrir l'ensemble des abonnés au téléphone mobile, cela passe par la proposition des services innovants qui inciteront les abonnés à adopter de nouveaux usages. Il est aujourd'hui certain que les services basés sur la localisation ou géodépendants se positionnent comme une valeur de différentiation pour les opérateurs, ils deviennent incontournables dans une logique de marché hautement compétitif. C'est la raison pour laquelle nous nous intéresserons à ces services dans ce chapitre car les applications de localisation vont dépendre de l'aptitude qu'aura ce service de fournir au serveur WAP une information de localisation fiable

Ce chapitre qui présente le WAP va avant tout s'appesantir sur les technologies (GSM, GPRS, UMTS) qui le supportent. La deuxième section de ce chapitre présente le réseau GSM du point de vue de l'architecture et des services offerts. La troisième section décrit la nome WAP, son histoire, sa spécification, son architecture et sa pile protocolaire. Cette section relève les limites du WAP sur GSM et évoque la nécessité du GPRS (on parle souvent de GSM phase 2). La quatrième section de ce chapitre présente l'architecture du GPRS et les services offerts. Nous faisons état dans la cinquième section des services et architecture relatifs au réseau de 3^{eme} génération qu'est UMTS. La sixième section de ce chapitre va s'intéresser aux services de localisation dans le réseau UMTS car ce service sera modélisé dans le chapitre 4 afin de rendre fiable l'obtention de l'information de localisation. Cette section présente entre autre les différents types de services de localisation, les différentes méthodes de localisation, l'architecture de ces services et un exemple d'application orientée localisation développée dans le cadre de ce travail de recherche.

2.2 Le réseau GSM

2.2.1 Présentation

Les réseaux de type GSM sont des réseaux complètement autonomes. Ils sont interconnectables aux RTCP (Réseaux Terrestres Commutés Publics) et utilisent le format numérique pour la transmission des informations, qu'elles soient de type voix, données ou signalisation. Les équipements spécifiques constituant le squelette matériel d'un réseau GSM (BTS, BSC, MSC, VLR et

HLR détaillés plus loin) dialoguent entre eux en mettant en oeuvre les mêmes principes que ceux utilisés dans le RNIS (Réseau Numérique à Intégration de Services) :

- Architecture en couche (couches 1 à 3 du modèle OSI) ;
- Utilisation des liaisons sémaphores (signalisation) ;
- Caractéristiques des liaisons identiques : vitesse codage MIC (Modulation par Impulsion et Codage).

2.2.2 Services

La téléphonie est le plus important des téléservices. Elle permet la communication entre deux postes mobiles et entre un mobile et un poste fixe, et ceci à travers un nombre quelconque de réseaux.
Le service d'appel d'urgence génère automatiquement un appel à destination d'un service d'urgence, quand l'utilisateur sélectionne la fonction appropriée. Le GSM propose également un service d'échange de messages alphanumériques courts que l'on appelle SMS (140 caractères au maximum). Il peut être mis en oeuvre pour l'émission depuis tous les terminaux capables d'émettre des messages alphanumériques vers un terminal GSM. Ce service est exploitable selon les modes point à point et point-multipoint.

2.2.3 Structure du réseau

Architecture

Comme on peut le voir sur la figure suivante, un PLMN (Public Land Mobile Network) de type GSM se présente sous la forme d'une structure hiérarchisée composée de quatre segments.

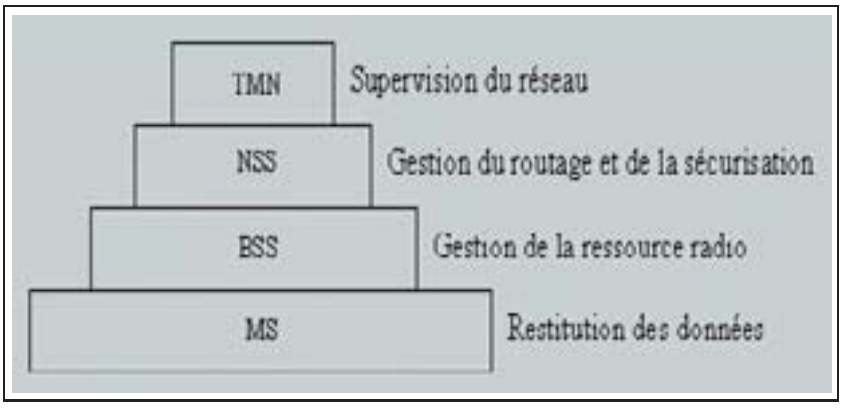


Fig. 2.1 – Structure hiérarchique dans GSM

Cette structure est constituée de quatre parties qui sont :

- le MS (Mobile Segment) qui est composé de terminaux portables,

- le BSS (Base Station Subsystem) regroupe les équipements assurants toutes les fonctions de gestion des aspects radio. Le détail des fonctions remplies par chaque sous-système est décrit dans les paragraphes suivants. Ce segment est composé de :
  - Une ou plusieurs BTS (Base Transceiver Station) qui assurent l'interface entre structures fixes et mobiles,
  - Un BSC (Base Station Controller) qui est le sous-système intelligent du BSS (analyse de données et prise de décision pour assurer la continuité de la communication dans la mobilité) ;
- le NSS (Network SubSystem) regroupe les sous-systèmes qui assurent des fonctions du niveau réseau (routage, interconnexion). Les équipements qui constituent ce segment sont :
  - Les bases de données HLR (Home Location Register) ; ce sous-système peut être considéré comme la mémoire centralisée du réseau ;
  - Les VLR (Visitor Location Register) qui peuvent être considérés comme des mémoires temporaires ;
  - Les commutateurs pour mobiles MSC (Mobiles-services Switching Center) assurent pour l'essentiel le routage et l'interconnexion avec le RTC (Réseau Téléphonique Commuté).
- le TMN (Telecommunication Management Network) regroupe les sous-systèmes qui assurent des fonctions de sécurisation, de supervision, de maintenance. Ce segment est constitué de :
  - L'EIR (Equipement Identity Register), qui a des fonctions de sécurisation ;
  - L'AUC (Authentification Centre), qui est une base de données utilisée pour la détection d'accès frauduleux ;
  - Les OMC (Operations and Maintenance Center), qui assurent des fonctions de configuration et de contrôle à distance. Le NMC (Network Management Centre) qui assure des fonctions de supervision du réseau.

Description fonctionnelle des différents sous-systèmes

La Base Transceiver Station (BTS) : ce sous-système est composé d'un ensemble d'émetteurs/récepteurs. Ce type d'équipement assure l'interface entre les mobiles et les structures fixes spécifiques au GSM. Ce sous-système est en charge :

- De la gestion du multiplexage temporel (une porteuse est divisée en 8 slots) ;
- Des mesures radio permettant de vérifier la qualité du service (mesures transmises directement au BSC) ;
- Des opérations de chiffrement ;
- De la gestion de la liaison de données au niveau 2 (données de trafic et signalisation) entre les mobiles et les structures fixes BTS (assuré

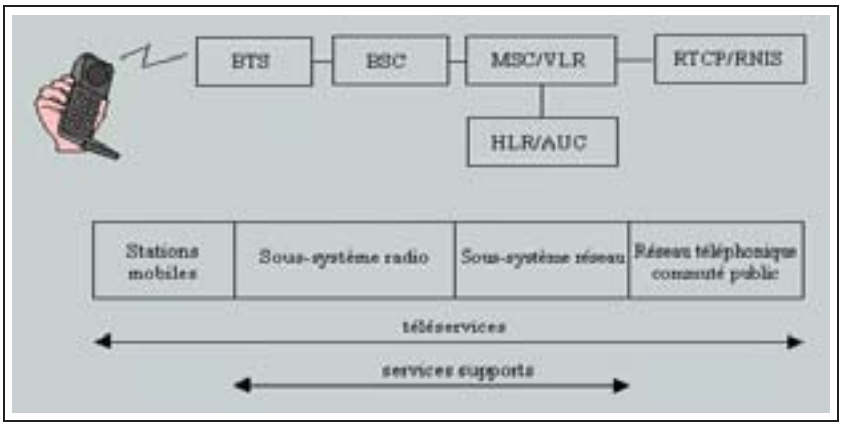


FIG. 2.2 – Structure d'un réseau GSM

par le protocole LAPDm) ;
- De la gestion des liaisons de trafic et signalisation avec le BSC (assuré par le protocole LAPD).

La capacité maximale typique d'une BTS est de 16 porteuses (16*7 = 112 communications simultanées). En zone urbaine, où le diamètre de couverture d'une BTS est très réduit, cette capacité peut descendre à 4 porteuses, soit 28 communications simultanées.

La Base Station Controller (BSC) : cet équipement assure l'interface avec le segment NSS (Network Sub-System) avec lequel il dialogue au travers de liaisons de type MIC (Modulation par Impulsion et Codage). Il assure également le contrôle des BTS qui dépendent de lui. Ses fonctions principales sont :

- L'allocation des canaux de communication ;
- Le traitement des mesures des niveaux d'émission BTS et mobiles ;
- La concentration de circuits routés vers le MSC ;
- La gestion des liaisons de communications.

Le Mobile-services Switching Center (MSC) : ce sous-système, que la norme couple à un VLR, a une fonction d'interconnexion avec le réseau public local. Il assure les fonctions permettant :

- L'interconnexion avec le réseau fixe ;
- Le routage après consultation du VLR associé (profil d'abonnement) ;
- La gestion de la mobilité pendant une communication.

Dans la pratique, ce sous-système intègre également les fonctionnalités de VLR qui sont décrites dans le paragraphe suivant.

Le Visitor Location Register (VLR) : ce sous-système s'interface avec le HLR, un MSC, d'autres VLR et l'AUC. Il assure des fonctions de base de données temporaire contenant les informations relatives aux terminaux présents et actifs (au moins en veille) dans son secteur de couverture. Il assure les fonctions permettant :

- L'acquisition des informations stockées au niveau du HLR lors de l'arrivée d'un nouvel abonné dans sa zone de couverture ;
- La mise à jour des informations de localisation contenues dans le HLR après contrôle de la validité de l'IMEI (International Mobile Equipement Identity) identifiant tout terminal GSM ;
- L'enregistrement des terminaux de passage ;
- L'authentification du terminal par contrôle du numéro IMEI affecté à chaque combiné.

Le Home Location Register (HLR) : cet équipement intègre la base de données nominale d'un PLMN. Il regroupe toutes les informations permettant de localiser et d'identifier tout terminal (sous tension) dont il a la charge. Il s'interface avec l'ensemble des VLR du PLMN et l'EIR. Il assure les fonctions permettant :

- La fourniture, sur demande d'un VLR, des informations relatives à un abonné dont il a la gestion ;
- L'acquisition d'informations (sur un abonné) issues d'un VLR, puis la mise à jour de la base de données qu'il contient ;
- L'acquisition des informations de chiffrement allouées à chaque abonné par l'AUC.

Les informations stockées sont :

- L'identité internationale de l'abonné (IMSI) ;
- Le numéro d'annuaire (MSISDN) ;
- La liste des services autorisés ;
- Le dernier numéro de VLR où l'abonné s'est inscrit.

Ces différentes informations sont :

- Soit centralisées sur une machine dédiée qui peut alors gérer plusieurs milliers d'abonnés ;
- Soit déportées sur les MSC/VLR, ce qui est souvent le cas en pratique.

L'identification du HLR concerné par une communication est faite par utilisation des numéros IMSI (Numéro d'abonné utilisé uniquement dans le cadre du réseau GSM) ou MSISDN (Numéro RNIS international d'une station mobile) du mobile.

Le terminal mobile : celui-ci est scindé en deux parties :

- Le combiné téléphonique identifié par un numéro unique : l'IMEI (International Mobile Equipement Identity) qui est l'identité internationale spécifique à chaque combiné. En pratique ce numéro n'est que

peu utilisé ;
- La carte SIM (Subscriber Identity Module) qui contient les informations suivantes :
  - Le numéro d'identification temporaire attribué par le réseau qui permet la localisation et qui est utilisé sur les canaux radio (TMSI Temporary Station Identity) ;
  - La liste des fréquences à écouter pour identifier la meilleure Station de Base ;
  - Les algorithmes de chiffrement.

Cette liste n'est pas exhaustive. D'autres informations sont stockées sur cette carte, tel que le code permettant de la débloquer : une carte SIM se bloque automatiquement après un certain nombre d'erreurs sur le code entré par l'utilisateur.

Cet ensemble permet d'accéder aux services d'un PLMN GSM. Cette découpe permet à l'usager d'utiliser n'importe quel terminal GSM car son identification complète est portée par la carte SIM. L'établissement d'une communication commence toujours par une phase d'authentification durant laquelle le réseau dialogue avec la carte SIM.

Le terminal mobile a pour seule interface les équipements de type BTS et ses fonctionnalités sont :
- La gestion de la liaison de données avec le BTS (protocole LAPDm) ;
- La surveillance périodique de l'environnement par des séries de mesure sur fréquences "balises" stockées sur la carte SIM ;
- La restitution des données vocales ou non (messagerie) destinées à l'abonné ;
- Les opérations de chiffrement.

Le concept de la mobilité : Le principe du handover

Les problèmes liés à la mobilité d'un terminal en communication, sont réglés conjointement par la structure fixe et le mobile. La décision d'effectuer un basculement de fréquence nécessaire au traitement d'un transfert intercellulaire (handover en anglais) reste toutefois à la charge des équipements fixes (MSC + BSC). Cette décision découle des traitements liés aux mesures sur le niveau de réception du mobile effectué par ce dernier (sur les fréquences balises environnantes) et transmises à la BTS nominale relayant la communication en cours.
Le principe repose sur :
- Les mesures faites par le terminal mobile et transmises au BSC courant ;
- La décision prise par le BSC d'effectuer un handover après identification d'une ou plusieurs cellules utilisables ; si plusieurs cellules sont

éligibles, le MSC détermine, en fonction des charges de trafic, la cellule la plus judicieuse à utiliser pour la communication ;
- La réservation d'un deuxième canal de trafic entre la nouvelles BTS et le mobile ;
- Un basculement effectué par le mobile sur réception d'une commande émise par le BSC.

Dans GSM, le handover s'effectue avec coupure de la communication (imperceptible pour l'utilisateur). C'est la structure fixe qui initialise la procédure en fonction des mesures effectuées par les mobiles en communication.

2.3 Le WAP : un premier pas vers l'Internet mobile

2.3.1 Historique

Comment a été crée le Wap ? - Ericsson, Motorola et la start-up Unwired Planet ont créé le Wap Forum en décembre 1997 afin de tenter d'établir des spécifications unifiées pour permettre aux terminaux sans fil, téléphones portables, *pagers*, et *Palm Pilots* l'accès à Internet.

L'utilisateur d'un terminal sans fil doit pouvoir recevoir, consulter et envoyer des mails, naviguer sur Internet et recevoir des informations sélectionnées sur Internet.
Le Wap Forum[1] a été créé pour définir les spécifications d'une norme internationale permettant l'accès à Internet via les terminaux mobiles. Cette norme définit les protocoles de connexion, de transmission, de navigation, etc.. Elle permet de traduire du texte provenant d'Internet dans un langage compréhensible par un téléphone mobile. Il existait déjà le HDML (Handheld Device Markup Language) assez proches du HTML, langage de programmation universel des sites internet.
Le Wap est devenu l'une des principales normes pour la présentation et la fourniture des informations et des services de téléphonie sur les téléphones mobiles et autres terminaux sans fil.
Le Wap Forum compte aujourd'hui plus de 250 membres, parmi lesquels on peut citer des fabricants de terminaux, des opérateurs, des éditeurs de logiciels, des fabricants d'infrastructures Wap : Alcatel, Bouygues Telecom, Cegetel/SFR, DeLaRue, Ericsson, France Telecom, Fujitsu, Motorola, Mitsubishi, NEC, Nokia, Nortel Networks, NTTDoCoMo, etc..
Les objectifs du Wap Forum sont :
- réunir les partenaires de l'industrie de la téléphonie mobile pour assurer l'interopérabilité et la croissance du marché cellulaire ;

[1] un forum n'est pas un organisme de normalisation mais il peut influencer la normalisation en soumettant ses spécifications pour qu'elles soient adoptées

- définir et promouvoir une norme d'accès à Internet pour les terminaux mobiles, le point de départ étant constitué des standards de type XML et IP ;
- contribuer au développement de services Internet pour les terminaux sans fil ;
- offrir du contenu Internet et des services de données avancés aux téléphones mobiles et autres téléphones sans fil ;
- créer une spécification de protocole sans fil globale qui fonctionne avec différentes technologies de réseaux sans fil ;
- permettre la création de contenu et d'applications qui soient utilisables sur différents réseaux et équipements sans fil ;
- inclure et étendre les normes existantes si nécessaire.

2.3.2 Les spécifications du WAP

Les premières spécifications Wap ont été rédigées dans le but de permettre aux fournisseurs potentiels de produits Wap de déployer des services et des produits rapidement. Mais il est nécessaire d'apporter des améliorations à ces normes, en particulier sur le thème de :

- *l'interopérabilité*, chaque client Wap équipé de téléphones, de *pagers* interagissant avec des serveurs Wap ne doit pas se préoccuper du fournisseur de ces serveurs ;
- *l'enrichissement des fonctions de sécurité*, nécessaire pour des applications comme le *e-business*, le *e-commerce* et assuré par l'utilisation de cartes SIM, cartes à puce et signatures numériques.

Les spécifications WAP 1.0 ont été publiées en septembre 1997 et ratifiées en avril 1998. Les premières versions des spécifications WAP 1.1 ont été publiées en mars 1999. Les spécifications WAP 1.1 finales ont été publiées à partir de juin 1999 et les premiers terminaux ne sont apparus qu'à partir de décembre 1999.

La version 1.2 des spécifications WAP intègrent des interfaces avec des cartes à puce, la technologie de *push*, de meilleures interfaces pour la facturation et des fonctions de localisation du terminal mobile pour de futures applications, a été finalisée en décembre 1999. Mais tout n'est pas stable. Ainsi, même si les spécifications sont normalisées, une même application ne fonctionnera pas de la même façon suivant le navigateur embarqué sur le terminal mobile.
La version WAP 2.0, plus adaptée, en termes de débit, aux technologies de transmission de données sans fil de type UMTS (avec des taux de transfert théorique de 2 Mbit/s), est disponible depuis septembre 2001.
De plus, WAP 2.0 supporte le *streaming* (permettant la transmission de vidéo), le téléchargement des fichiers MP3 (utilisés pour la musique), ainsi que le langage XHTML (eXtensible HyperText Markup Language).
Enfin, WAP 2.0 permet de reconnaître le type de terminal sans fil sollici-

tant des données, et en particulier la taille de son écran ou le logiciel de navigation utilisé, et d'adapter l'affichage à la taille de l'écran du terminal récepteur.
Les normes WAP sont développées de façon à compléter les normes existantes : les spécifications WAP ne spécifient pas comment les données sont transmises sur l'interface radio mais elle sont rédigées de façon à donner la possibilité d'utiliser les standards de transport des données existants. Le WAP Forum travaille conjointement avec le *World Wide Web (W3C)* afin d'aligner les normes WAP avec le monde Internet.

2.3.3 Architecture du WAP

Les éléments de l'architecture

Le WAP permet d'accéder depuis un terminal compatible à des contenus adaptés de sites préexistants ou développés spécifiquement pour ce standard. Pour accéder à un service, le téléphone mobile via le bowser envoie des requêtes au format WML (Wireless Markup Languge), un dérivé du HTML spécialement développé en tenant compte des caractéristiques des réseaux mobiles. Il a pour but de réduire le plus possible la quantité d'information transitant sur le réseau GSM, vu sa lenteur. La requête est ensuite transmise à une passerelle qui se charge de récupérer les informations demandées à partir d'un serveur internet traditionnel ou de préférence d'un serveur dont les contenus ont déjà été optimisés avec le langage WML.

Si le contenu vient d'un site WEB classique, la passerelle se charge de le filtrer pour le traduire dans la langue du browser. Par suite, l'information est retransmise par la passerelle, via le réseau GSM, jusqu'au téléphone mobile de l'abonné.

La transmission des informations entre le terminal qui intègre un microbrowser et le réseau (ou le bouquet e-services) s'effectue, soit par l'intermédiaire d'une passerelle qui assure la transformation des requêtes HTTP classiques, soit directement par requêtes *Wireless Transaction Protocol (WTP)* sur un serveur d'application. La figure 2.3 montre l'échange d'information entre un terminal mobile et un serveur.

La passerelle : c'est un élément clé dans les échanges. Elle fournit une connexion directe entre le réseau mobile et les serveurs d'application. Elle joue le rôle de client de tous ces serveurs et des technologies comme le GPRS, les services de messages courts et les données à commutation de circuit permettant d'y accéder. Elle inclut les fonctionnalités suivanes :

- un *protocole* qui traduit les requêtes WAP en requêtes HTTP ;
- des *codeurs et décodeurs de contenu :* le codeur de contenu traduit le contenu WEB en un format codé compact afin de réduire la taille et le nombre des paquets traversant le réseau de données sans fil.

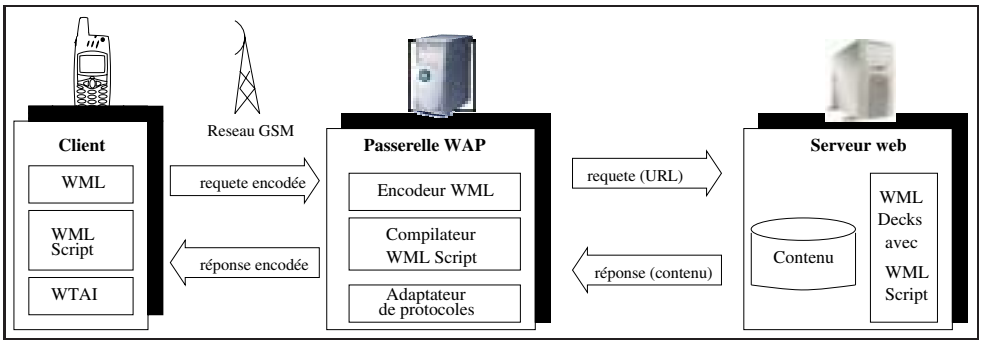


FIG. 2.3 – Architecture logique du WAP

Les documents WML sont structurés en plusieurs unités appelées cartes. Les services sont créés en laissant l'utilisateur naviguer d'une carte à une autre d'un ou plusieurs documents WML (*deck*). La gestion des services est basée sur le principe des jeux de cartes (appelé *deck* de cartes). Un deck est envoyé du réseau vers le terminal de l'utilisateur quand il entre une demande de services. En fonction de la capacité du terminal, les *decks* peuvent être sauvegardés dans le terminal pour une utilisation ultérieure.

2.3.4 Piles de protocoles WAP

L'architecture se compose de couches successives pour les applications, les sessions, la sécurité et le transport des informations. La navigation sur un téléphone portable est réduite à sa plus simple expression.
La technologie WAP n'a rien de révolutionnaire, elle s'appuie sur une pile de protocoles inspirée de celle de TCP/IP qui prend en compte des contraintes liées aux terminaux mobiles telles que :

- faible bande passante de transmission ;
- limitation de la taille de l'écran ;
- limitation de la résolution de l'écran ;
- limitation de la puissance de calcul ;
- faible autonomie des batteries ;
- etc..

La figure 2.4 compare les protocoles utilisés dans le monde Internet et ceux utilisés dans le monde WAP. Elle permet de positionner les protocoles WAP par rapport aux protocoles Internet.

Wireless Application Environment

Les outils permettant de développer une application sont regroupés dans le *Wireless Application Environment*, l'environnement de développement d'application WAP qui est, pour l'essentiel composé des éléments suivants :

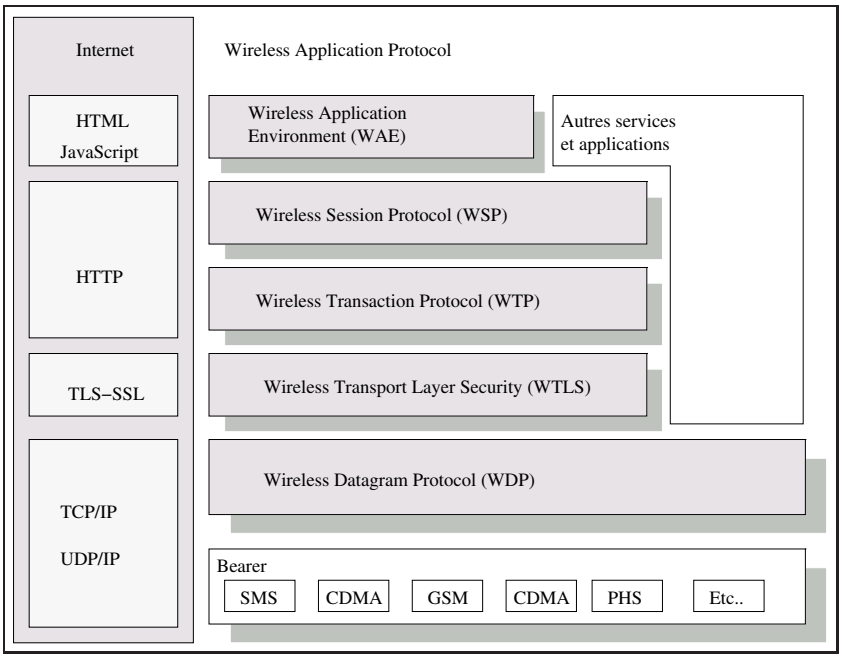


FIG. 2.4 – Pile de protocoles

- le WML, un langage de description des pages ;
- le WML Script, un langage de script ;
- le Wireless Telephony Application (WTA), un ensemble de spécifications qui ont pour but de réaliser l'interaction entre les services de données et les services voix. WTA est un agent intégré au mobile ;
- WBMP, un format graphique ;
- le support des standards vCard et vCalendar pour la transmission respectivement d'adresses et de rendez-vous.

Wireless Session Protocol

Dans l'architecture WAP, la couche session est définie par WSP. La version actuelle de WSP définit des services essentiellement utilisables par un micro-browser. Cette version est nommée WSP/B (B pour Browser).
Cette version de WSP propose une interface à deux services de session. Le premier est un service connecté, qui opère au-dessus de la couche transaction (WTP). Le second est un service non connecté, qui agit au-dessus du service paquet (WDP). Il peut être utilisé si vous ne désirez pas avoir une confirmation de l'arrivée des données.
W S P/B fournit les fonctionnalités suivantes :

- fonctionnalité sémantique de HTTP 1.1 ;
- session longue ;

- gestion des reprises de session et de mise en attente avec migration de session ;
- protocole de négociation ;
- gestion du *push*.

Wireless Transaction Protocol

WTP fonctionne au-dessus du service de paquet (WDP) et est orienté transaction. Il fournit les fonctions suivantes :

- Trois classes de service de transaction :
  - Classe 0 : les requêtes sans accusé de réponse, ni message de réponse (cela permet à une application de transmettre des données sans savoir si elles sont arrivées).
  - Classe 1 : les requêtes avec accusé de réponse mais sans message de réponse (l'application émettrice a besoin d'être sûre que son message est bien arrivé).
  - Classe 2 : les transactions requête-réponse.
- Transaction asynchrone. Cela permet aux destinataires de la requête de renvoyer la réponse dès que possible.
- Mécanisme de concaténation et de confirmation de plusieurs unités de données pour réduire le nombre de messages, ainsi que les temps de réponse et l'encombrement de la bande passante.

Wireless Transport Layer Security

WTLS est un protocole de sécurité basé sur le standard TLS *(Transport Layer Security Protocol)*, plus connu sous l'acronyme SSL *(Secure Sockets Layer)*. WTLS a été conçu pour supporter les réseaux à faible bande passante. WTLS fournit les fonctions suivantes :

- Intégrité des données : le WTLS garantit que les données échangées entre le mobile et la passerelle ne sont ni altérées, ni changées, ni corrompues ;
- Vie privée : le WTLS assure que les échanges sont privés et ne peuvent être compris par un tiers qui aurait intercepté le flux de données ;
- Authentification : Le protocole intègre des mécanismes pour établir une communication où le mobile et la passerelle peuvent être identifiés par des échanges de certificats.
- Non-répudiation : le WTLS garantit que la transaction ne va pas être dénoncée par l'une des parties impliquées. Cela peut être géré par des certificats clients et serveurs ou plus simplement par la saisie d'un code utilisateur et d'un mot de passe par utilisateur. Cette pratique est la plus courante.

Wireless Datagram Protocol

WDP est un protocole de transport. Il fonctionne au-dessus des capacités de transfert de données des réseaux mobiles. Il peut s'appuyer sur des réseaux divers tels que SMS, GSM, GPRS, USSD.
Son rôle est de fournir la même couche d'abstraction aux différents services orientés données de tous les types de réseaux disponibles. C'est WDP qui permet aux couches session , transaction, sécurité et application (respectivement WSP, WTP, WTLS, et WAE) d'être indépendantes du réseau.
Le protocole WDP est orienté datagramme (sans connexion). Il est adapté à tout réseau sans fil afin de fournir un transport de datagrammes générique, facilitant l'utilisation du WAP n'importe où. Le protocole WDP est remplacé par UDP quand il est utilisé sur la couche réseau IP.

2.3.5 Les limites du service WAP et nécessité du GPRS

Le wap lockage

Le WAP *lockage* est une technique de verrouillage utilisée par l'opérateur qui place son bouquet de services en environnement propriétaire de sorte que tout utilisateur soit obligé d'utiliser le bouquet de services ou le portail et la passerelle d'accès d'un opérateur. Les terminaux sont verrouillés sur la page d'accueil de l'opérateur : au démarrage, les utilisateurs "tombent" obligatoirement sur le portail de l'opérateur ne proposant que ses services. Le verrouillage des téléphones mobiles empêche l'installation de toute autre page de démarrage. Certes, il est possible d'accéder aux services ne faisant pas partie du bouquet de l'opérateur, mais il faut pour cela taper l'adresse internet en toutes lettres ce qui est long (donc coûteux) et difficile sur un téléphone mobile.

Les limites héritées du GSM

- **Terminaux** : il est vrai que la configuration des terminaux évoluent très vite, cependant les limites suivantes avaient été observées du point de vue des terminaux GSM :
  - l'écran : de petite taille, la faible résolution (au mieux 100*100 pixels), monochrome, rend difficile la consultation des informations ;
  - le clavier offre un nombre limité de touches, ce qui rend les informations difficiles à saisir. Pour composer un message, il faut souvent frapper les mêmes touches plusieurs fois ;
  - les batteries, même de dernière technologie, ne sont pas assez performantes pour permettre de rester en communication une journée. Il est difficile d'envisager des séances de travail en nomade ;
  - les débits faibles (limité à 9,6 kbit/s) ;

  - la puissance de traitement des terminaux : insuffisance de mémoire et de la capacité des microprocesseurs.
- **Adaptation des sites** : nécessité d'adapter les sites Internet aux caractéristiques du WAP. C'est-à-dire d'adapter le contenu des sites Internet à l'ergonomie réduite des téléphones mobiles. Les téléphones mobiles doivent être petits pour les utilisateurs nomades et les applications doivent être simples afin d'améliorer la productivité des travailleurs nomades. Les bouquets de services actuels offrent peu de services qui ne puissent être remplacés par un simple coup de fil.
  La mise en service de véritables moteurs de recherche WAP permettant un accès plus large au "WEB" devait permettre aux développeurs de sites de proposer des services inédits, plus imaginatifs et attractifs.
- **Lacunes logicielles** : le WAP comporte des lacunes au niveau logiciel. Les outils de développement, par exemple, impliquent de coder les pages WML à la main. De plus, les différents navigateurs inclus dans les nouveaux téléphones interprètent encore les pages WML de façon inégale. Une simple conversion des pages WEB en pages WAP n'est pas satisfaisante dans la majorité des cas.
- **Connexion complexe** : les connexions sont parfois longues, difficiles et lentes ; les déconnexions sont fréquentes, la vitesse de transfert des données est faibles et la connectivité avec les assistants personnels et les PC n'est pas toujours possible. Les transferts de données sont facturés à la minute, que la connexion fonctionne correctement ou non, l'utilisation du WAP devient donc très coûteuse pour l'utilisateur.
- **Amélioration du WAP** : d'après les spécialistes, le véritable décollage du WAP dépend de trois facteurs : l'arrivée des terminaux plus ergonomiques, le développement des contenus et enfin l'augmentation du débit.

Nécessité du GPRS, service réel de données

La première étape de l'accès au WAP consiste à se connecter. Là, tout se passe comme sur internet depuis un ordinateur. Le WAP est basé sur la transmission IP et le protocole HTTP et utilise une transmission de données binaires pour faciliter la compression. Le téléphone compose le numéro de téléphone du fournisseur de services à internet (souvent l'opérateur de réseau mobile) et se connecte à une passerelle.Une fois la connexion établie, le terminal mobile va chercher à afficher la page d'accueil.
Quelques modifications sont nécessaires afin que l'usage des nouvelles applications Internet sur les terminaux mobiles devienne naturel pour les utilisateurs :

- une personnalisation du contenu, des informations pour l'utilisateur selon son type de terminal mobile ;
- une fiabilité de l'infrastructure (plus de sécurité est nécessaire) ;

- les constructeurs doivent mettre sur le marché des terminaux mieux adaptés ;
- les nouvelles applications et services Internet doient être développées en WAP ;
- de nouveaux débits plus adaptés sont nécessaires. Les débits proposés par le GPRS permettront une nette amélioration de la perception des services WAP par les utilisateurs. Ansi, avec le GPRS, le WAP sera carectérisé par un délai de connexion réduit à environs 2 secondes.

2.4 Le réseau GPRS

2.4.1 Présentation

Le GPRS ne constitue pas à lui tout seul un réseau mobile à part entière, mais une couche supplémentaire rajoutée à un réseau GSM existant. Il peut donc être installé sans aucune licence supplémentaire. Ceci signifie que tous les opérateurs qui disposent d'une licence GSM peuvent faire évoluer leur réseau vers le GPRS. L'ART n'a d'ailleurs pas fait d'appel d'offre pour le GPRS alors qu'elle en a fait pour l'UMTS.
De plus, le GPRS utilise les bandes de fréquences attribuées au GSM. C'est à dire une bande dans les 900 MHz, une autre dans les 1800 MHz et enfin une troisième pour les USA, dans les 1900 MHz. Les opérateurs GSM actuels ont de fait un quasi monopole sur le GPRS, ce qui n'est pas le cas pour l'UMTS.

Le GPRS, appelé aussi GSM 2+, repose sur la transmission en mode paquet. Ce principe déjà, retenu par exemple pour le protocole X.25, permet d'affecter à d'autres communications les "temps morts" d'une première communication (attente d'une réponse à une requête Internet par exemple).
Conçu pour réutiliser au maximum les infrastructures GSM existantes, le déploiement du GPRS nécessite la mise en place d'une infrastructure réseau basée sur la commutation de paquets et l'introduction de passerelles pour s'adosser aux réseaux GSM existants.
Cette technologie, capable de fournir des débits par utilisateur allant jusqu'à 115 kb/s (contre 9,6 kb/S pour le GSM), offre des fonctionnalités intéressantes :

- plusieurs canaux peuvent être alloués à un utilisateur ;
- ces mêmes utilisateurs peuvent partager un même canal ;
- le débit est indépendant des liens montant et descendant.

2.4.2 Services/Possibilités/Applications

L'accès immédiat et fiable

Le GPRS offre un accès immédiat. Le mode de fonctionnement du GPRS et son mode de facturation au volume de données transmises, permet de laisser le canal de transmission ouvert en permanence. Ainsi, pour télécharger un e-mèl par GPRS on économise, par rapport à une connexion par GSM ou RTC, lors de la première connexion, le temps d'initialisation du modem, soit 30 secondes environs. Sur les autres e-mèls, l'avantage est encore plus flagrant, les téléchargements se font immédiatement, sans numérotation préalable alors qu'en GSM il faut recommencer la procédure de numérotation pour chaque consultation.

Mode connecté ou accès virtuel

Le premier avantage du GPRS est de permettre une meilleure utilisation des ressources radio et techniques. Alors que le GSM actuel fonctionne en mode "connecté", appelé également mode "circuit", le GPRS utilise pour sa part le mode de connexion virtuel. En mode "virtuel", les ressources sont partagées. Le canal de transmission n'est jamais affecté à un utilisateur unique, mais partagé entre un certain nombre d'utilisateurs. Chaque utilisateur en dispose lorsqu'il en a besoin et uniquement dans ce cas. Le reste du temps elles sont disponibles. Le mode "connecté" quant à lui correspond au fonctionnement d'une ligne GSM ou encore d'une ligne téléphonique standard. Il consiste à établir un lien physique entre deux points ou deux correspondants.

2.4.3 Architecture du réseau

La figure 2.5 présente l'architecture du réseau GPRS ainsi que les interfaces entre les différents noeuds et les éléments du réseau GSM existant.

L'utilisateur est équipé de son terminal TE (Terminal Equipment) qui correspond à l'ensemble formé du MT (Mobile Terminal) et de la carte SIM de l'utilisateur.

Le terminal est raccordé au sous-système radio (le même que celui de GSM excepté quelques modifications), appelé BSS. Ce sous-système radio est connecté au sous-système réseau GSM (MSC/VLR, HLR, EIR) via l'interface A et au sous-système réseau GPRS via l'interface Gb par, respectivement, le MSC/VLR et le SGSN.

certains éléments du réseau GSM sont représentés sur la figure 2.5 afin de souligner l'intégration nécessaire du réseau GPRS dans le réseau GSM existant.

Le sous système GPRS se compose des éléments principaux suivants :

- des SGSN : Serving GPRS Support Node, serveur d'accès au service GPRS (équivalent au MSC) ;
- des GGSN : Gateway GPRS Support Node, routeur IP connectant le réseau GPRS et un réseau externe de commutation par paquet (IP ou X.25).

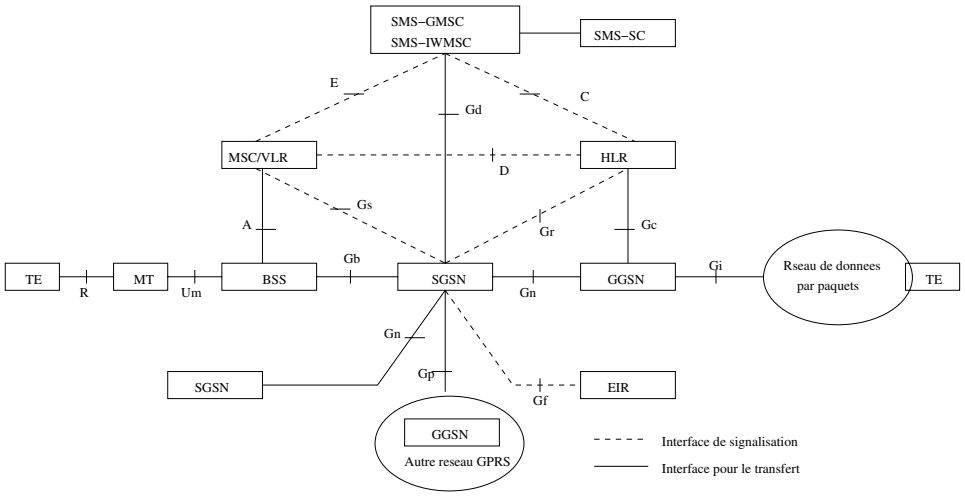


FIG. 2.5 – Architecture du GPRS

2.5 Le réseau UMTS

2.5.1 Présentation

L'UMTS est la version européenne définie par l'ETSI (Institut Européen de Normalisation des Télécommunications) de la troisième génération des services mobiles (3G). Il devrait délivrer des débits compris entre 384 kb/s à 2 Mb/s.
En fait, cette norme est un membre de famille du projet IMT-2000 (International Mobile Telecommunication System 2000) défini par l'UIT (Union Internationale des Télécommunications). Celui-ci a pour but de normaliser les systèmes de télécommunications mobiles de troisième génération qui assureront l'accès radioélectrique à l'infrastructure mondiale des télécoms, dans un contexte mondial d'itinérance. Il doit faire intervenir aussi bien les systèmes satellitaires que les moyens terrestres desservant les usagers fixes et mobiles des réseaux publics et privés.
Ainsi, les réseaux UMTS constitueront les systèmes de télécommunications mobiles et sans fil de troisième génération, capables d'offrir au grand public des services de type multimédia à débit élevé. Voici les objectifs que le parlement européen assigne au projet :

- Pour les services :
  - Capacités multimédias et mobilité sur une très grande étendue géographique ;
  - Accès efficace à Internet, aux intranets et aux autres services basés sur le protocole IP ;
  - Transmission vocale de grande qualité, comparable à celle des réseaux fixes ;
  - Portabilité des services dans les environnements UMTS différents ;
  - Fonctionnement en mode GSM / UMTS à l'intérieur, à l'extérieur et dans des endroits extérieur éloignés, sans solution de continuité, permettant une itinérance totale entre les réseaux GSM et entre les éléments terrestres et satellitaires des réseaux UMTS.
- Pour les terminaux :
  - Terminaux GSM / UMTS bimodaux et à deux bandes, si approprié ;
  - Terminaux UMTS bimodaux terrestres / satellite, si approprié.

2.5.2 Attribution de licence

D'après l'institut européen de l'audiovisuel et des télécoms, "la plupart des pays doivent respecter les directives de l'Union Européenne visant à permettre l'ouverture commerciale des réseaux de 3^{eme} génération au 1^{er} janvier 2002". Ainsi les opérateurs devront acquérir une licence UMTS dans un pays donné pour pouvoir utiliser une bande de fréquence appropriée et développer sa propore infrastructure.
Certains pays, comme le Royaume-Uni, l'Allemagne et les Pays-Bas, se sont prononcés en faveur du système des enchères. D'autres, comme la France, le Portugal, la Finlande, l'Espagne et le Danemark, ont choisi le système de soumission comparative (beauty contest) avec droit d'entrée pour la France.

2.5.3 Services

Services traditionnels

De nombreux services orientés données sont supportés par le GSM. En particulier les évolutions du GSM, telle que le GPRS (115 kb/s), permettra une première étape vers la transmission haut débit, et vers d'autres services tels que le courrier électronique, le télépaiement, le transfert de fichiers, l'accès Internet. Toutefois, l'UMTS (384 kb/s pour tout le monde en mode mobile et 2 Mb/s en situation "fixe") fournira un meilleur compromis capacité/coût.

Services émergents

A moyen ou long terme, l'UMTS s'adressera au grand public (d'ailleurs, il existe déjà des réseaux pré-commerciaux d'infrastructure mobile UMTS

dans plusieurs régions de France). Dans ce cadre, la grande majorité des services qui seront proposés alors est encore inconnue. Ces nouveaux services répondront certainement à trois exigences :

- Contenus multimédia (exemples : jeux, loisirs) ;
- Mobilité ;
- Valeur ajoutée (il faut que le grand public soit prêt à payer le surcoût de ces services).

2.5.4 Architecture et structure de l'UMTS

L'architecture du réseau

Le réseau UMTS est divisé en deux parties : le coeur du réseau et le réseau d'accès (figure 2.6) :

- *le réseau d'accès* est la partie du réseau qui gère l'interface et les ressources allouées sur l'interface radio. Un autre rôle important du réseau d'accès est la gestion de la mobilité de l'usager. La couverture radio étant constituée de cellules de taille variable, le réseau d'accès doit être capable de faire passer l'usager d'une cellule à une autre en cours de communication. Il s'agit de la fonction de *handover*.
- *le coeur du réseau* est la partie du réseau qui gère l'ensemble des abonnés et les services fournis aux abonnés. Le réseau coeur est responsable de l'établissement de la communication et assure la liaison entre le réseau UMTS et les réseaux extérieures.

  La figure 2.6 présente l'architecture générale d'un réseau UMTS utilisant le réseau d'accès UTRAN[2]. L'ensemble des constituants et des interfaces du réseau GSM 2+ (en fait GPRS) ont été repris dans l'architecture du réseau coeur UMTS.
  Le réseau d'accès se compose quand à lui du RNC (Radio Network Controller), équivalent du BSC des réseaux GSM, et du NodeB, équivalent de la BTS.

La figure 2.7 (partie haute) présente la structure du réseau UMTS pour les appels en mode circuit. Les couches de transport TCAP, SCCP et MTP utilisés dans le réseau coeur sur les interfaces C, E et F sont identiques au GSM. La couche applicative MAP de l'UMTS est une évolution de la couche MAP du GSM par rapport aux nouveaux services définis dans le cadre de l'UMTS.
La figure 2.7 (partie basse) présente le plan de transmission des données usagers des services en mode paquet. Comme pour les services en mode circuit, les couches transport du réseau coeur entre SGSN et GGSN sont inchangées.
La signalisation appartenant à la couche NAS du réseau, c'est à dire les

[2]Universal Terrestrial Radio Access Network

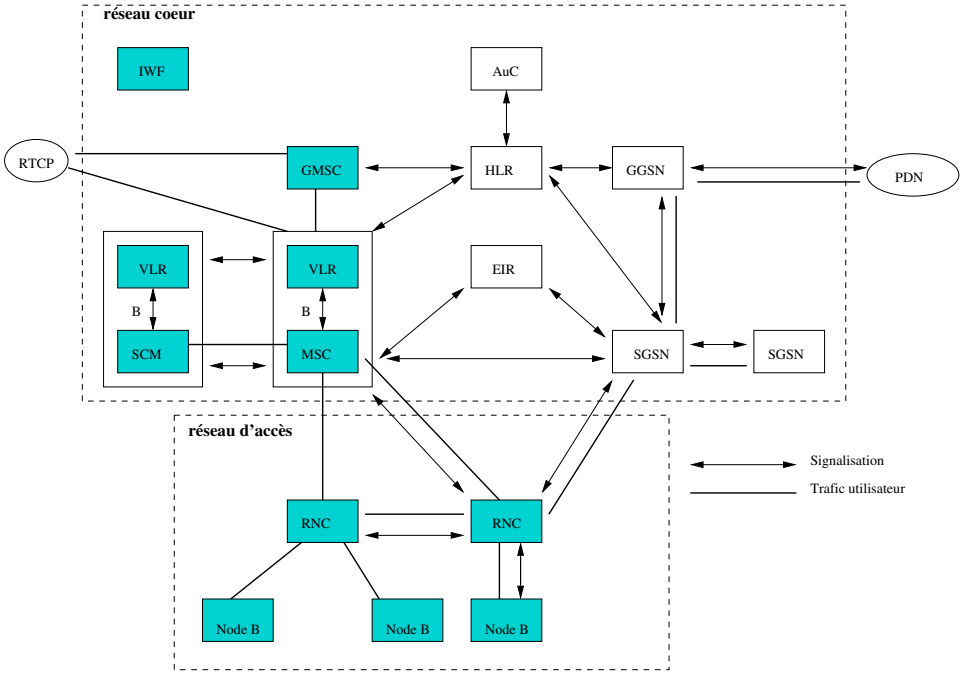


FIG. 2.6 – Architecture du réseau UMTS

couches CC (Call Control) et MM (Mobility Management), pour les appels circuit, et SM (Session Management) et GMM (GPRS Mobility Management), pour les appels paquet sont également identiques à celle utilisées en GSM, à quelques évolutions près.

2.6 Les services de localisation

La localisation (voir l'intérêt de ces services dans l'introduction à ce chapitre) est la capacité d'un système de communication à déterminer la position géographique d'un terminal, mobile ou non. Une fois localisé, il est possible d'envoyer à l'utilisateur du terminal des informations relatives aux commerces, aux services, aux lieux remarquables qui existent près de l'endroit où il se trouve.
Dans l'architecture générale d'une requête nécessitant la localisation (figure 2.8) nous avons :

- 1 : Le mobile envoie une requête à la passerelle
- 2 : La passerelle transforme celle-ci en requête HTTP et l'envoie au serveur (client-LCS)
- 3 : Afin de localiser le terminal, le serveur envoie une requête de localisation au serveur de localisation (serveur-LCS)
- 4 : L'information de localisation ainsi obtenue est renvoyée au client-

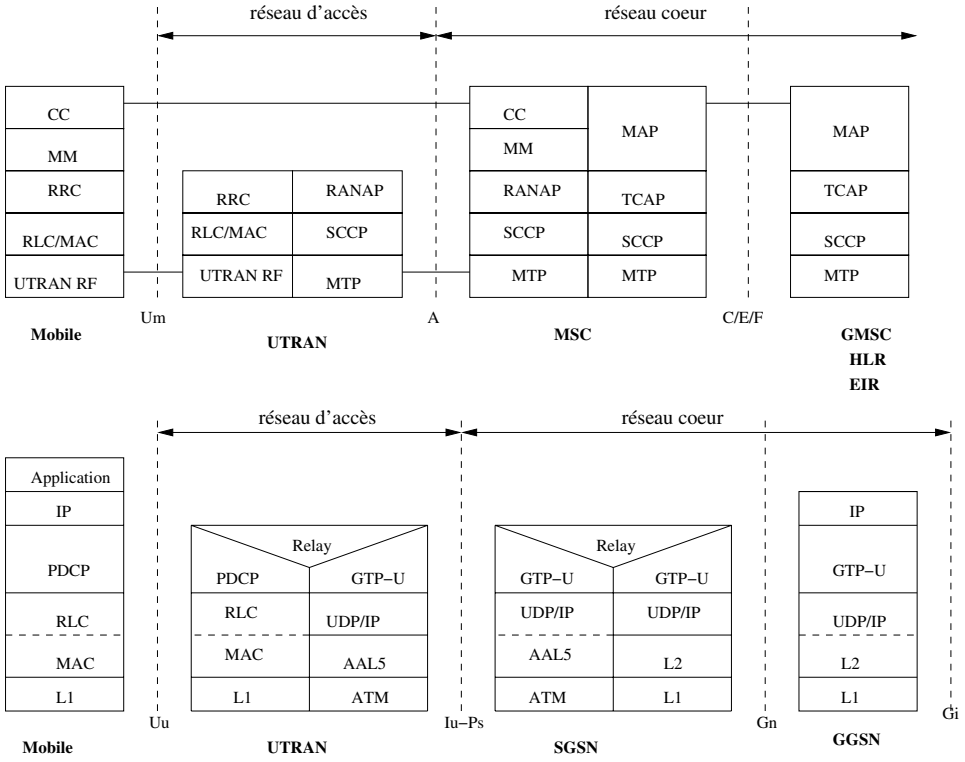


FIG. 2.7 – Architecture en couche en mode circuit (haut) et paquet (bas)

LCS

– 5,6 : Les données relatives à la position sont renvoyées au terminal via la passerelle

Les services orientés localisation sont variés mais quatre d'entre eux représentent les plus gros potentiels de revenus :

2.6.1 Les services d'information

Trouver un service donné le plus proche, disposer d'une information de trafic routier, bénéficier d'une aide de guidage dans une ville inconnue, tels sont des exemples des services géo-dépendants. Ceux-ci s'expriment :

– Dans une relation de type "Business to Consumer" (entreprise à particulier), ce nouveau média d'information est exploité par les entreprises. Grâce à la localisation des abonnés, celles-ci peuvent cibler les abonnés ayant choisi de souscrire à cette option de service dans leur campagne de publicité ou de promotion de produits de proximité.

– dans une relation de type "Consumer to Business" (particulier à entreprise), l'abonné fait lui même une demande de localisation afin de recevoir une information précise en fonction de son environnement. Il

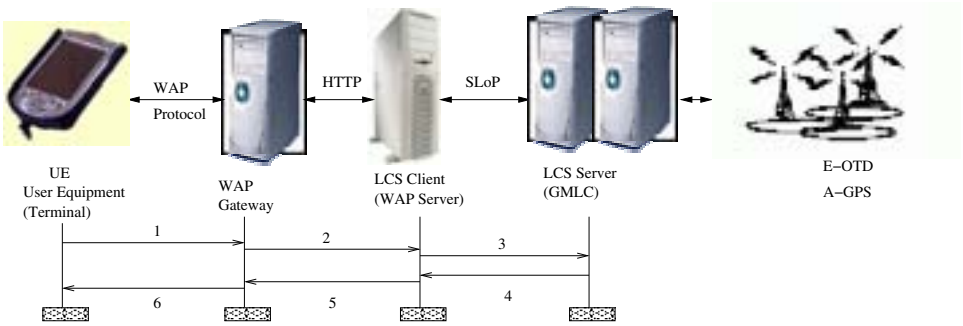


FIG. 2.8 – Déroulement d'une requête WAP nécessitant la localisation du terminal

peut s'agir d'une demande d'informations de proximité (restaurant, station d'essence...).
– dans une relation de type "Consumer to Consumer" (particulier à particulier), l'abonné peut localiser des amis, des membres de sa famille ou encore plus largement des membre d'une communauté à laquelle il adhère (sport, musique...).

2.6.2 Les services d'assistance

Il s'agit des services d'assistance publics ou privés en usage piéton ou automobile. Par exemple, les services publics d'urgence font appel à l'intervention des forces de sécurité publiques (pompiers, SAMU..).

2.6.3 Les services de suivi de flottes de matériels ou d'individus

Ce sont pour l'essentiel des services de type "Business to Business" (inter-entreprises) qui mettent en oeuvre des applications de gestion de flottes et qui consistent à localiser une ressource pour en optimiser l'exploitation, la gestion ou en assurer la sécurité.

2.6.4 Les services opérateur

L'expoitation des informations de localisation permet d'améliorer des applications de services dans des domaines divers comme la planification de réseau, la qualité de service, l'optimisation des ressources radio et la tarification.

2.6.5 Les méthodes de localisation

Il existe plusieurs méthodes permettant le calcul de la position d'un terminal mobile.

Global Positioning System - GPS

Le GPS utilise un ensemble de satellites pour localiser l'utilisateur. Avec le GPS, le terminal recueille l'information brute des satellites. Cette information brute sera traitée par le terminal ou par le réseau afin de recupérer la position courante. La précision varie de 5 à 40m.

Assisted Global Positioning System - AGPS

Cette méthode est basée sur la constellation des satellites GPS et l'ajout des serveurs de référence dans le réseau cellulaire et permet d'améliorer la précision.

Enhanced Observed Time Difference, E-OTD

Une station doit recevoir un signal synchrone de la part du mobile, la différence de temps de transmission entre le mobile et deux BTS décrit une hyperbole. Avec trois stations on peut estimer la position du mobile grâce à l'intersection des hyperboles. L'exactitude de la position est de 125m, mais à la différence du GPS cette méthode ne dépend pas de la clarté du ciel.

CGI-TA Cell Global Identity Timing Advance

Les zones couvertes par chaque BTS sont décomposées en cellules, on utilise donc l'identité de chaque cellule pour localiser l'utilisateur. Cette technologie est très peu coûteuse car elle utilise les terminaux déjà existants. La précision varie entre 10 et 50m selon la taille de la cellule.

TOA Time Of Arrival

Le signal entre la station et le mobile est continu, de ce fait, la position du mobile peut être connue en calculant le temps de propagation du signal radio, en forçant le BTS à envoyer un handover[3] au mobile.

2.6.6 Architecture standardisée d'un réseau de localisation

Les nouveaux éléments de réseau nécessaire à la localisation sont de trois types :

[3]un handover ou transfert intercellulaire permet d'assurer une continuité des appels même si les utilisateurs changent de cellule

- Le GMLC (Gateway Mobile Location Center) : il est responsable de l'interface vers le monde extérieur, notamment les "LCS-client" (LCS : service de localisation), qui sont à l'origine des requêtes de localisation. C'est ce centre qui reçoit les requêtes de localisation, authentifie le client, et vérifie qu'il est bien autorisé à demander la localisation d'un abonné. Le GMLC est également chargé de transmettre vers le réseau le niveau de qualité de service requis (précision, temps de réponse...), et de convertir les résultats du positionnement dans le format souhaité ;
- Le SMLC (Serving Mobile Location Center, Centre de localisation de mobiles de desserte) : qu'il soit intégré au RNC (Radio Network Controller - contrôleur de réseau radio) ou déporté dans un élément séparé, son rôle est de déterminer la position du mobile, c'est à dire ses coordonnées géographiques et éventuellement une incertitude. Il est libre de choisir la méthode de positionnement en fonction de la qualité de service demandée par le GMLC et des capacités du mobile. C'est également le SMLC qui reçoit les informations de couverture et de planification cellulaire lorsque la connaissance géographique du réseau est nécessaire ;
- La LMU (Location Measurement Unit, unité de mesure de positionnement), uniquement pour OTDOA[4] : le rôle des LMUs est d'assister le SMLC dans les mesures de synchronisation des stations de base. Elles sont soit intégrées au Node B (LMU de type B), soit disséminées dans le réseau (LMU de type A). On considère qu'il faut une LMU pour 3 à 4 stations de base.

Aussi, les éléments de réseau existants doivent également être modifiés. Le MSC doit notamment être capable de vérifier auprès du HLR (Enregistreur de localisation nominal) qu'un LCS-client est autorisé à localiser un abonné en fonction entre autre, de son identité et du profil de l'abonné. Pour les méthodes de localisation nécessitant que le mobile soit en communication, le MSC doit également être capable d'activer une communication en le notifiant ou non à l'abonné. Le RNC est également affecté, puisqu'il doit intégrer les fonctions SMLC ou bien piloter le SMLC. Enfin, les Node B doivent intégrer les LMU type B, et pouvoir mesurer les RTT[5].
La figure 2.9 représente l'architecture globale d'un réseau UMTS avec les éléments nécessaires à la localisation telle qu'elle est définie dans la norme 3GPP.

2.6.7 Le service applicatif

Dans l'environnement WAP, la couche WAE (Wireless Application Protocol) fournit un cadre de spécification d'application, les services applica-

[4]Observed Time Difference of Arrival
[5]Round Trip Time

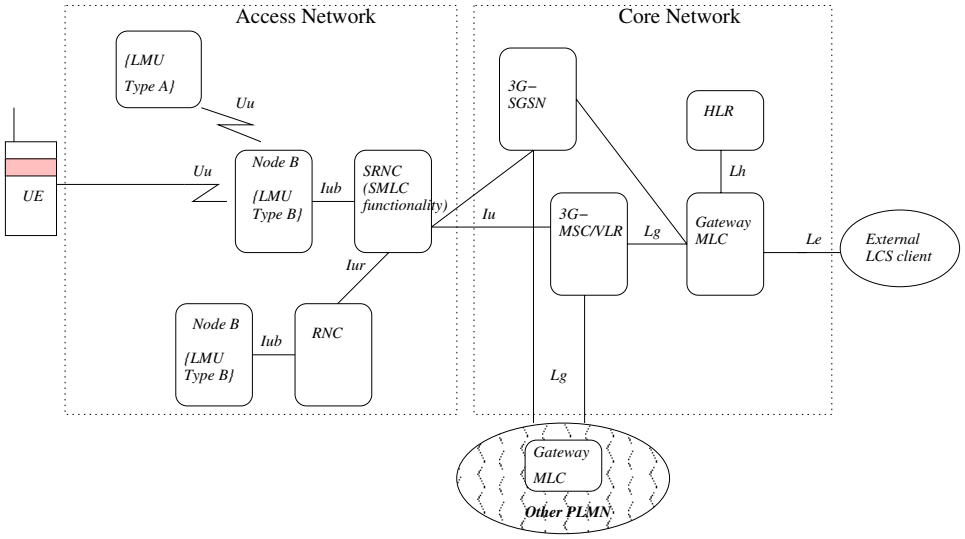


FIG. 2.9 – Architecture d'un réseau de localisation conforme au standard

tifs correspondent à l'exécution des programmes interagissant directement ou indirectement avec les utilisateurs finaux munis de leur mobile. Dans le cadre du projet PLATONIS[6], nous avons développé en WML (Markup Wireless Language) une application LOW (Local info On Wap) qui est un service d'information géo-dépendant de type "customer to business" qui prend donc en compte la localisation des abonnés pour fournir des services comme l'itinéraire, la proximité, le trafic etc..

- Les itinéraires aident les abonnés à planifier leurs voyages, à gérer leur temps et à atteindre leur destination en décrivant au mieux l'itinéraire et le temps de voyage selon l'état du trafic courant ;
- Les services de proximité permettent aux abonnés de rechercher et/ou de repérer dans leur voisinage les points tels que : station, hôpitaux, parkings, arrêts de bus, musées, monuments, services publics, etc.. ;
- les cartes de trafic aident les abonnés à éviter l'encombrement en affichant l'information à jour du trafic.

La figure 2.10 présente les différents écrans du mobile lors de la navigation. Pour ce faire, l'abonné charge l'url de l'application dans son terminal. Il verra par la suite apparaitre la page d'accueil de l'application LOW suivi du menu principal qui propose l'accès aux différents services offerts tels que : la proximité, l'itinéraire, le trafic, etc..

La figure 2.11 montre comment l'application se présente dans un terminal

[6]PLAte-forme de validatiOn et d'expérimentation multI-protocoles et multi-Services

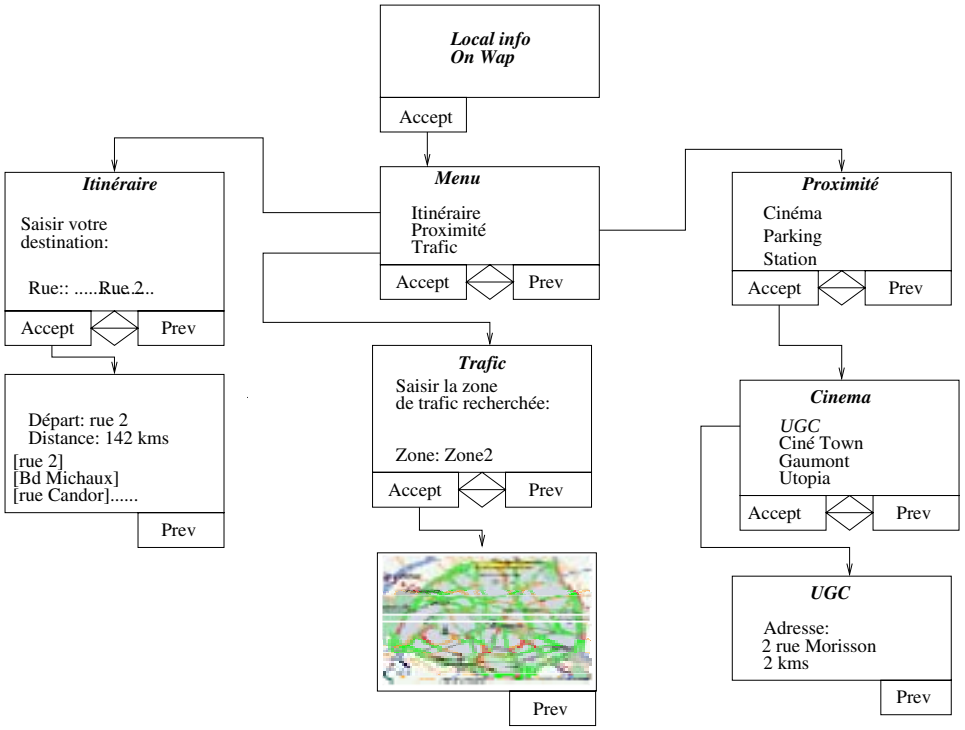


FIG. 2.10 – Diagramme de défilement d'écran de l'application

(ici le simulateur de Nokia). En sélectionnant l'itinéraire à partir du menu, une boîte de dialogue nous permet de renseigner notre destination. Après validation de la destination saisie, il nous est demandé de préciser notre moyen de transport pour ensuite recevoir un détail du trajet à parcourir rue par rue ou métro par métro selon le moyen de transport choisit. Plus de détails sur l'application sont donnés dans l'annexe A.

2.7 Conclusion

Ce chapitre a présenté l'intérêt grandissant de faire de l'Internet dans un contexte de mobilité. Le WAP permet en effet d'offrir aux possesseurs de téléphones mobiles des services plus avancés que les messages courts mais hérite des faibles débits permis par GSM (9,6 kbit/s).
Le WAP a été nommé à tort "Internet Mobile". Il correspond plutôt à une première expérience d'utilisation sur un téléphone mobile de quelques applications équivalentes à celles rencontrées sur sur Internet. Il doit plutôt être considéré comme un moyen pour les utilisateurs de tester un service de données s'appuyant sur un réseau cellulaire, c'est à dire un avant goût du GPRS. Il permet aussi aux opérateurs d'observer le marché des données sur

FIG. 2.11 – Itinéraire

réseaux cellulaires et d'analyser le potentiel de tels services avant d'investir dans les réseaux de 3^{eme} génération tels que l'UMTS.

Les services basés sur la localisation ou géo-dépendants se positionnant comme une valeur de différentiation pour les opérateurs, ils deviennent incontournables dans une logique de marché hautement compétitif. Ce chapitre a également présenté le concept et l'architecture des services LCS (service de localisation) qui seront modélisés dans la suite de ce travail de recherche.

Chapitre 3

Le test : Etat de l'art et stratégie de test des protocoles/services mobiles

3.1 Introduction

Lors de la conception d'un système informatique, une phase de *validation/vérification* est nécessaire pour s'assurer que celui-ci est bien conçu et fournit le service attendu. Le système supposé valide fait généralement l'objet d'une *norme* qui sera par la suite *implantée* par différents constructeurs. Ainsi, une *implantation* est un produit industriel commercialisé. Les propriétés d'une implantation (par exemple des grandeurs physiques, des fonctionalités données) peuvent être déterminées au moyen de tests.

3.1.1 Les tests de système

Les tests de système sont définis suivant les objectifs qu'ils visent ; on distingue les types de tests suivants :

- **tests de conformité** : l'objectif de ce type de tests est de vérifier que l'IST (implantation sous test) est conforme, c'est-à-dire qu'elles satisfait la spécification du système considéré. Comme nous le verrons par la suite, ce type de test a été défini en détail par les organismes de normalisation ISO (Organisation Internationale de Normalisation) et ITU (International Telecommunication Union).
- **tests de robustesse** : l'objectif, dans ce cas, est de déterminer le comportement de l'IST dans un environnement "hostile", c'est-à-dire dans des conditions anormales de fonctionnement. Par exemple, on teste l'implantation en lui envoyant des interactions erronées. De telles situations (erronées) ne sont pas décrites par la spécification du système. Elles sont très variées et nombreuses (voir infinies). Leur inclusion ex-

plicite dans la spécification pourrait rendre cette dernière encore plus volumineuse.
- **tests de performance** : l'objectif de ce type de tests est de mesurer certains paramètres de performance de l'IST. Des paramètres de performance couramment rencontrés sont le débit maximal, les détails de transmission et le nombre de connexions supportées en parallèle.
- **tests de d'interopérabilité** : ce type de tests a pour objectif de vérifier que plusieurs systèmes informatiques interopèrent, c'est-à-dire qu'ils sont conformes et aptes à communiquer en vue de rendre le service auxquel ils sont destinés. Ce type de tests n'est pas encore formalisé par l'ISO. Mais son intérêt est croissant car les tests de conformité ne sont pas exhaustifs en pratique et ne suffisent donc pas pour garantir l'interopérabilité.

La majorité des tests normalisés à ce jour sont les tests de conformité. Des travaux de synthèse [7, 69] ont présenté une bonne introduction générale au test de conformité.

3.1.2 Pourquoi utiliser des méthodes formelles ?

Les protocoles ISO définissent des systèmes relativement complexes. Les comportements possibles de ces systèmes sont très nombreux, voire infinis. La production manuelle de test est :
- *un travail de longue haleine* : ceci est dû à l'aspect volumineux et complexe des spécifications. Lorsque celles-ci sont écrites textuellement (de centaines de pages quelquefois), il est impossible d'en faire un traitement automatique, d'où l'intérêt de moyens formels de spécification.
- *Un travail d'expert* : seul un expert ayant une bonne connaissance du système est capable d'écrire un jeu de tests pertinent.
- *Un travail répétitif* : d'un système à un autre, "le même travail" de génération répété : lire la spécification, l'interpréter, écrire des tests. Sur la base d'un formalisme donné de spécification, l'on peut développer des algorithmes de génération de tests réutilisables, quelque soit le système spécifié dans le formalisme considéré.
- *Un travail difficile à évaluer* : la mesure de *couverture de tests* [85, 11, 18], devrait être utilisée pour évaluer un ensemble de tests donnés. Quelle propriété a-t-on effectivement testée, quel taux de fautes a-t-on couvert ? Cette mesure est difficile à obtenir à partir d'une suite empirique de tests et d'une spécification textuelle informelle.

3.1.3 Classification des méthodes formelles de test

Les points précédents donnent des arguments en faveur des méthodes formelles de test et les organismes de normalisation sont favorables au développement de telles méthodes. Les méthodes que nous qualifiont de *formelles*

utilisent, en général, une description des protocoles par des systèmes à transitions (automates, systèmes de transitions étiquetés) ou par les TDF (technique de description formelle) normalisées (Estelle [77], Lotos [26], SDL [87]) qui sont plus riches en pouvoir d'expression (notamment pour les contraintes liées aux données). La majorité de ces TDF est basée sur un modèle à transition donné et la génération de tests se fait en général à partir de ce modèle sous-jacent (cf. annexes de la norme $Z500$ [88]).
Sur la base de travaux menés par le groupe de travail ISO/IEC JTC 1/SC 21 Projet 54 ITU TQ.8/10, d'une part et de techniques récentes de génération de tests, d'autre part, nous faisons une classification de méthodes formelles de tests de conformité [48] en trois grandes classes (ou tendances), que nous appelons *méthodes utilisant les automates d'états finis (MAEF)*, *méthodes utilisant la théorie des testeurs canoniques (MTTC)* et *méthodes orientées objectif de test (MOOT)*. La norme $Z500$ formalise des concepts qui sont essentiels dans le processus de génération de test et qui sont mis en oeuvre dans ces classes de méthodes selon des démarches différentes.

Dans la première classe de méthodes, la génération de tests est guidée par la structure (états, transitions associées) de l'AEF de la spécification. Le test consiste alors à vérifier que l'IST à une structure équivalente [19, 62, 47, 73, 30]. Dans la deuxième classe de méthodes, on fait abstraction de la structure interne de l'entité ; on considère plutôt le comportement externe, que l'on caractérise par une propriété dite *relations d'implantation* et qui permet de comparer deux processus (par exemple une spécification, une implantation, généralement modélisées par des systèmes de transition étiquetées) [9, 23, 51, 82]. Un processus de test est défini pour vérifier cette relation d'implantation.

Ces deux premières classes de méthodes visent à établir une relation entre le comportement global de l'IST et celui de la spécification. Pour des systèmes ayant des comportements infinis (ou qui explose en taille), il est impossible de réaliser des tests exhaustifs. Il faudrait donc utiliser une méthode plus pratique consistant à choisir certains comportements particuliers que l'on voudrait tester, ceci permettant de limiter l'explosion combinatoire inhérente à toutes ces méthodes de génération de test ou de vérification.

La troisième classe de méthodes s'intéresse à la génération de tests à partir d'une propriété spécifique, appelée *objectif de test*. Dans la norme ISO/IS 9646 [78], un objectif de test décrit de manière informelle l'objectif spécifique d'un cas de test. Il permet ainsi de sélectionner un cas de test permettant de tester des comportements particuliers parmi l'ensemble (*a priori* infini) des comportements possibles de l'IST [84]. Indépendamment des spécificités des méthodes de test, la bibliographie montrera que cette classification en trois tendances, à un caractère chronologique : les méthodes utilisant les AEF ont été les premières à être développées, car les premiers systèmes (notamment

les protocoles de communication) étaient modélisés avec des automates. Ensuite, la nécessité de construire une théorie adaptée au test de conformité a motivé le développement de la deuxième classe. Enfin, le besoin de produire des tests concrets pour des systèmes réels a conduit à imposer des méthodes pragmatiques de génération de tests. On peut établir un lien théorique entre ces différentes classes de méthodes, voire entre les modèles qu'elles utilisent. Les systèmes sur lesquels portent nos travaux de recherche (les systèmes basés sur les protocoles/services du WAP et les technologies UMTS/GSM/-GPRS) mettent en relation des éléments hétérogènes, si la conformité de chacun des éléments de la chaîne est assurée, nous ne pouvons garantir l'interfonctionnement du système global. C'est ainsi que dans la section 3.3 nous abordons le test d'interopérabilité, sa nécessité, sa problématique et le contexte de recherche.
Dans la section 3.4, nous proposons une stratégie en vue de tester la conformité des composants et l'interopérabilité de notre système global.

3.2 Le test de conformité

Nous présentons d'abord les modèles utilisés dans ce mémoire, à savoir des systèmes à transition. Ensuite, nous rappellerons quelques concepts définis dans le cadre des normes. La norme $ISO9646$ [78] définit une méthodologie pour test de conformité (terminologie, démarche générale). Des articles de synthèse présentent les principaux concepts de cette norme [7, 69]. La norme $Z500$ apparaît comme un complément orienté vers les méthodes formelles.

3.2.1 Les modèles à transitions

Historiquement, les systèmes de transitions étiquetées ont été essentiellement utilisés pour modéliser des processus concurrents [63]. Les automates d'états finis (ou machines de Mealy) [32, 36] ont permis de modéliser des systèmes tels que les circuits électroniques se synchronisant par des entrées et des sorties. Comme nous le verrons par la suite, les automates dits à entrées-sorties (IOSM) [56, 66] se placent entre les deux précédents modèles.

Définition 3.1 *Un STE (système de transitions étiquetées) M est un quadruplet (S(M), L(M),T(M), s0(M)) où :*

1. *S(M) est un ensemble dénombrable d'états,*
2. *L(M) est un ensemble dénombrable d'actions observables,*
3. *$T(M) \subseteq S(M) \times [L(M) \cup \tau] \times S(M)$ est l'ensemble des transitions ($\tau \notin L(M)$ est appelée action interne ou inobservable),*
4. *$s_0(M) \in S(M)$ est l'état initial de M.*

Déterminisme d'un STE

Un STE est dit déterministe si aucun état n'admet plus d'un successeur par action, c'est à dire : $\forall \mu, \forall s, (s \stackrel{\mu}{\Rightarrow} s_1 \; et \; s \stackrel{\mu}{\Rightarrow} s_2) \Longrightarrow s_1 = s_2$.

Définition 3.2 *Un AEF (automate d'états finis) peut être défini par un quintuplet (S,I,O,T,s0) où :*

1. *S est un ensemble fini d'états,*
2. *s0 est l'état initial,*
3. *I est un ensemble fini d'entrées,*
4. *O est un ensemble fini de sorties,*
5. *$T \subseteq S \times I \times O \times S$ est l'ensemble des transitions. Une transition (s, i, o, s') signifie que si l' AEF est dans l'état s et qu'il reçoit l'action d'entrée (ou tout simplement l'entrée) i, il emet l'action de sortie o et passe dans l'état s'.*

La différence essentielle avec un *STE* (fini) réside dans le fait que les transitions d'*AEF* sont définies par des couples à *entrée/sortie* [48]. Des propriétés relatives aux *AEF* peuvent également être définies sur les *STE* : *complétude*, *déterminisme*, *connexité*, *minimalité*.

Définition 3.3 *Un IOFSM (input Output Finite State Machine) est un six-uplet (S, X, Y, s_0, δ, λ) [89] où :*
- *S est l'ensemble des états,*
- *X est l'ensemble des messages que le système peut recevoir, l'alphabet d'entrée,*
- *Y est l'ensemble des messages que le système peut émettre, l'alphabet de sortie*
- *s_0 l'état initial,*
- *$\delta : S \times X \rightarrow 2^S$ est une fonction appelée fonction de transfert,*
- *$\lambda : S \times X \rightarrow 2^Y$ est une fonction de sortie.*

3.2.2 Un cadre formel, la norme Z500 [88]

Les rubriques qui suivent présentent quelques concepts intervenant dans une démarche formelle de test.

Spécifications, Implantation, Modèles

Une *spécification* est la prescription d'un comportement à l'aide d'un moyen formel de description. On note *SPECS*, l'ensemble des spécifications. Une *implantation* est une entité matérielle et/ou logicielle. L'implantation possède des interfaces à travers lesquelles elle peut communiquer avec son environnement. On note *IMPS* l'ensemble des implantations. La spécification

est un objet formel, tandis que l'implantation est physique. Afin de pouvoir lier de tels objets (de nature différente) par la conformité, on suppose (*hypothèse dite de test*) que chaque implantation $IST \in IMPS$ peut être modélisée par un élément $m_{IST} \in MODS$, où $MODS$ est un formalisme donné (un ensemble de *Modèles*), par exemple, STE ou AEF ou $IOFSM$. Il est plus naturel que $SPECS$ et $MODS$ correspondent au même formalisme.

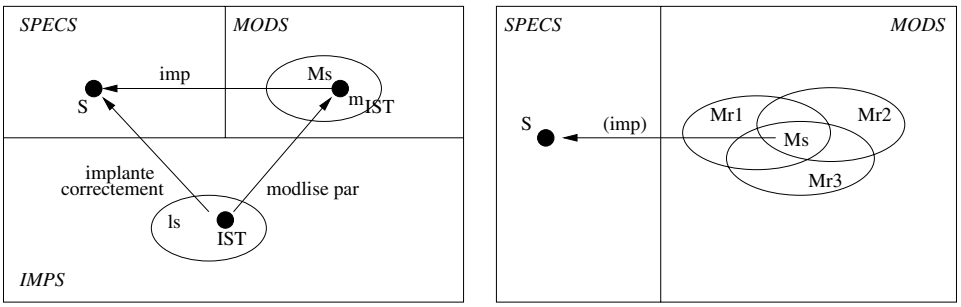


FIG. 3.1 – Expression de la conformité entre spécification et implantation

Conformité

La *conformité* est une propriété observable sur le comportement d'une implantation. Elle est relative au fonctionnement spécifié. La conformité peut être définie au moyen d'une relation binaire ou d'un ensemble de critères de conformité :

- *Relation de conformité imp*, de signature $imp \subseteq MODS \times SPECS$. Une implantation IST est dite conforme à une spécification S selon imp, si l'on a m_{IST} imp S (qui peut aussi s'écrire $imp(m_{IST}, S)$). La relation imp est un moyen d'exprimer la notion de *validité* entre une spécification S et une implantation IST. Dans la figure 3.1, Ms représente l'ensemble des modèles m tels que m imp S.
- *Critères de conformité REQS*, la conformité peut également être définie au moyen d'un ensemble de critères [38]. Un critère de conformité est une propriété requise pour qu'une implantation soit conforme. Dans la figure 3.1, M_{r1} (resp. M_{r2}, M_{r3}) représente l'ensemble des modèles qui satisfont le critère $r1$ (resp. $r2, r3$). Les critères doivent être exprimés dans un langage donné à priori. $REQS$ désigne alors l'ensemble des critères que l'on peut exprimer dans ce langage.

Le test de conformité peut être qualifié de test *boîte noire* [60], c'est-à-dire que l'on ne connaît pas la structure interne de l'implantation du système à tester, et que l'on obtient les tests à partir de la description des fonctions du

système. Pour cette raison, ce type de test est aussi appelé test fonctionnel. Il est opposé au test structurel ou test *boîte blanche*, où la connaissance de la structure interne de l'implantation à tester conditionne la génération des tests. Le test *boîte blanche* semble plutôt réservé aux logiciels, puisque l'on dispose des sources du programme à tester.

Des relations d'implantation pour exprimer la conformité

Les relations d'implantation que nous présentons ont été souvent utilisées [9, 23, 51, 66, 82] car elles s'accordaient très bien au concept de conformité et de test de l'ISO (ces relations sont basées sur une sémantique de trace, ou de refus). La relation *conf* a initialement été proposée par Brinksma [9] pour exprimer formellement la notion de conformité.

– *La relation conf*

$$conf(I,S) \Longleftrightarrow \begin{cases} \forall \sigma \in Tr(S),\ \forall A \subseteq L(S) \cup L(I), \\ \text{Si } \exists s \in S(I) \mid s_0(I) \stackrel{\sigma}{\Rightarrow} s \text{ et } \forall a \in A, s + a \\ \text{alors } \exists s' \in S(S) \mid s_0(S) \stackrel{\sigma}{\Rightarrow} s' \text{ et } \forall a \in A, s' + a \end{cases} \tag{3.1}$$

Dans cette relation d'implantation, les traces de "référence" sont celles de la spécification. Après avoir exécuté une trace quelconque σ autorisée par la spécification, tout ensemble A d'actions refusé par l'implantation I est nécessairement refusé par S, après que S ait également exécuté σ. En d'autres termes si on exécute l'implantation comme si elle était la spécification, elle ne conduit jamais à un blocage non prévu par la spécification.
conf est réflexive mais et non transitive sauf si on lui ajoute des propriétés d'inclusion de traces.

– *La relation ioco*
La relation *ioco* est l'une des plus courantes en test de conformité. Elle est notamment utilisée dans l'outil TGV [44] ainsi que dans des outils industriels tels que TestComposer [46] de Telelogic.
ioco(I,S) : une implémentation I est conforme à sa spécification S si après toute séquence σ de S, incluant éventuellement des blocages[1], les sorties de I sont incluses dans celles prévues par S.

Architecture de test

Le test de conformité est un moyen expérimental visant à vérifier qu'une implantation satisfait une propriété de conformité donnée. Il est mis en

[1]Le blocage est caractérisé par l'entrée dans un état où aucune action n'est possible

oeuvre dans une *architecture de test* (figure 3.2) et constitué de :

- *L'environnement de test* : représente un contexte dans lequel l'IST est plongé (par exemple, représentation d'une pile de couches de protocoles) ;
- *Le testeur* : c'est le testeur qui envoie les entrées au système à tester, collecte les sorties et doit être capable d'émettre un verdict ;
- *Le* PCO[2] : le testeur communique directement avec ce contexte via les $PCOs$ et indirectement avec l'IST via le contexte de test ;
- PAI[3] : la communication entre l'IST et le contexte de test se fait à travers des $PAIs$. La figure 3.2 est une vue abstraite de l'architecture de test qui peut être configurée de différentes manières [38, 69].

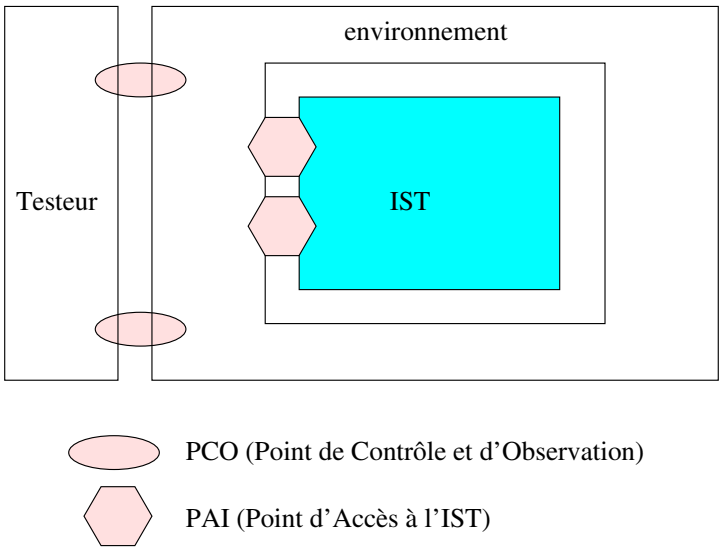


FIG. 3.2 – Architecture de test

Exécution des tests

- *Suite de test, cas de test* : le testeur est formalisé par l'ensemble des comportements à exécuter pendant l'expérience de test. Un tel ensemble est appelé *suite de tests*. Chaque élément de cet ensemble est appelé *cas de test* et permet de tester un aspect spécifique de l'IST.
- *Verdict* : l'exécution d'un cas de test conduit à un verdict (*Fail, Pass ou Inconclusive*) qui est une affectation de résultat à l'expérience de test. Le verdict *Fail* signifie que l'IST n'est pas conforme à sa spécification. Un verdict *Pass* signifie que la séquence d'interaction correspond à la séquence d'interaction de la spécification. Enfin, le verdict *inconclu-*

[2] point de contrôle et d'observation
[3] points d'accès à l'implantation

sive est utilisé lorsqu'on ne peut pas conclure, i.e. établir un des deux verdicts précédents. Les cas de test sont exprimés dans un langage de test donné. $TESTS$ désigne l'ensemble des tests que l'on peut exprimer dans ce langage. Une suite de tests est par conséquent un élément de $\wp(TESTS)$, où $\wp(X)$ désigne l'ensemble des parties de X.

Génération de suites de tests

Dans le processus de génération de suites de tests, une suite de tests est produite à partir d'une spécification formelle donnée. On définit la *génération* de test par une fonction *gen* :

$$gen : SPECS \rightarrow \wp(TESTS)$$

Propriétés : étant donné une spécification S et une suite de tests $T = gen(S)$, T peut être qualifiée d'exhaustive ou de correcte. Pour une suite de tests T idéale, une implantation est conforme si et seulement si elle ne révèle pas d'erreur lorsqu'on lui applique T. Mais la nature infinie des comportements considérés entraîne qu'une telle suite de tests n'est pas toujours applicable dans la pratique.

- *Exhaustivité* : un test est exhaustif si une implantation qui ne révèle pas d'erreur pendant ce test est effectivement une implantation conforme.
- *Correction* : un test est correct si une implantation qui révèle une erreur pendant ce test est nécessairement non conforme.
- *Complétude* : un test est complet s'il est exhaustif et correct.

Le langage des tests

L'ISO a normalisé un langage de test, qui est une notation, TTCN[4] [1]. Cette dernière permet de décrire un ensemble de tests qui devront être exécutés sur le système. En utilisant TTCN, une suite de test peut être spécifiée, il s'agit là d'une collection de cas de tests. TTCN est un langage abstrait en ce sens où il est indépendant du système de test réel.

3.2.3 MAEF : méthodes basées sur les AEF

Dans cette classe de méthodes, les systèmes considérés sont *modélisés* par des AEF ($MODS$ est instancié par AEF) et le test est basé sur la structure de l'AEF de spécification. Le but est de vérifier que l'IST implante bien une structure (états-transition) équivalente. La relation de conformité *imp* qui guide la génération de tests est basée sur une sémantique de traces et, comme nous le verrons par la suite, les automates considérés doivent être

[4]Tree and Tabular Combined Notation

déterministes [48]. Une classification d'erreurs est faite sur la base de la structure d'AEF. Toute implantation exempte de telles erreurs est considérée comme possédant les mêmes traces que l'AEF de spécification [55, 65]. Les méthodes considérées consistent alors à calculer des séquences aptes à détecter des erreurs sur la base de cette structure d'AEF. Ces séquences constituent les suites de tests à appliquer.

Nous présentons d'abord les modèles d'erreur (ou de faute) servant de base pour définir ce qu'est une implantation *conforme*, puis nous introduisons les méthodes de *génération* de test ayant le pouvoir de détecter ces erreurs. Nous expliquons ensuite les conditions sous lesquelles ces méthodes peuvent être appliquées, tout en rappelant leur lien avec la norme Z500.

La conformité basée sur la couverture de fautes

Dire qu'un système est conforme revient à dire que ce dernier ne contient pas d'erreur. Un test donné aura pour objectif de rechercher les erreurs possibles d'une implantation. Une classification de comportements erronés, appelés "modèles de fautes", est proposée par Bochmann [8]. Pour les AEF, les principales erreurs identifiées dans la littérature sont les erreurs dites *de sortie* (action de sortie erronée) et *de transfert* (état d'arrivée erroné). D'autres types de fautes peuvent être définis. Par exemple, les fautes dites *multiples* : l'ajout, la suppression d'état ou de transition, sont une combinaison de fautes de transfert et/ou de sortie [73, 68].

Génération de cas de test et couverture

Les principales méthodes connues sont la méthode du tour de transition (TT) [62] et les méthodes nécessitant l'existence de séquences particulières : la méthode W [19], la méthode de la séquence de distinction (DS :Distinguishing Sequence) [47] et la méthode d'entrées-sorties uniques (UIO :Unique Input Output) [72].

Applicabilité, lien avec la norme Z500

La section précédente a illustrée les MAEF et leur aptitude à détecter les erreurs. Ces méthodes sont répandues certes, mais elles ne s'appliquent qu'à des AEF ayant des propriétés spécifiques :

– *Déterminisme, équité* : Ces méthodes ne s'appliquent qu'à des systèmes (des AEF) déterministes pour lesquels, dans un état donné, la réception d'une entrée détermine, sans ambigüité, les entrées suivantes que le système est autorisé à recevoir. Une implantation non déterministe n'est pas contrôlable et le testeur ne sait pas, à priori, comment elle choisit ses chemins. Toutes les méthodes qui génèrent des suites finies de tests font donc l'hypothèse suivante : "Au bout d'un certain nombre de fois où les séquences de tests seraient répétées, l'IST aura

mis en oeuvre tous ses comportements possibles". Cette hypothèse également connue sous le terme *d'équité*, est similaire à celle faite, pour les systèmes non déterministes, par Milner [67].
- *Connexité, réinitialisabilité* : L'AEF doit être *connexe* afin de pouvoir atteindre tout état à partir de l'état initial. Afin de pouvoir exécuter une autre séquence commençant par l'état initial, l'on doit pouvoir y revenir depuis l'état où l'on se trouve pendant le test. L'AEF doit donc être *réinitialisable*. Pour cela, il suffit qu'il soit fortement connexe, ou qu'il possède un mécanisme de réinitialisation ("reset").
- *Minimalité, complétude* : Si l'objectif est de trouver des séquences permettant de distinguer les états entre eux, il va de soi que l'AEF considéré doit être sous forme minimale. Autrement, des états pourraient être équivalents, auquel cas, aucune séquence (ou ensemble de séquences) ne permettrait de les distinguer. De plus, l'on doit connaître le comportement qui est prévu dans un état donné, quelque soit l'entrée reçue. Selon cette entrée, le comportement varie d'un état à un autre (exemple, aller dans un état puits ou non) et il doit être complètement spécifié.

Lien avec la norme : L'ensemble $MODS$ est instancié par AEF. La propriété de conformité *imp* considérée est une équivalence de traces, car les MAEF visent à établir une équivalence entre les états de la spécification et ceux de m_{IST} (modèle de l'IST).
Des propriétés de déterminisme, minimalité et complétude assurent aux MAEF de générer des tests *corrects*. Par ailleurs, comme l'ensemble des tests générés est fini, la propriété d'exhaustivité ne peut être satisfaite que sous l'hypothèse d'équité d'une part, et sous l'hypothèse d'une connaissance à priori du nombre d'états de l'IST, d'autre part.
Les hypothèses contraignantes citées ci-dessus limitent en effet l'utilisation de ces méthodes qui n'ont pas été réellement appliquées à des cas complexes et n'ont pas donné lieu à de véritables outils.

3.2.4 Méthodes utilisant la théorie des testeurs canoniques

Les MTTC utilisent une démarche pouvant se résumer comme suit : (i) une spécification S est considérée ; (ii) une propriété (appelée *relation d'implantation)* R est définie pour exprimer la notion de conformité à S ; $R(I, S)$ signifie que I est une implantation conforme à S ; (iii) un testeur T, dit *canonique* est définie pour tester $R(I, S)$.

Pendant la communication entre T et I, T doit être capable de détecter si I viole la relation R. Les systèmes considérés sont modélisés par des STE ($MODS$ est représenté par STE).

Applicabilité, lien avec la norme

L'intérêt des MTTC est qu'elles offrent un cadre plus rigoureux, voire théorique pour exprimer ce qu'est la conformité et le test de conformité. Mais les testeurs canoniques ont généralement des exécutions infinies, car les relations d'implantation qu'ils testent sont généralement infinies (exemple la relation $conf$ introduite précédemment). Ce qui induit alors la problématique suivante :

- Comment choisir un ensemble fini de séquences de test ?
- Comment exécuter toutes ces séquences pendant une campagne de test ?

Lien avec la norme : l'ensemble $MODS$ est représenté par STE. La relation de conformité *imp* à été représentée par *conf* et la démarche de génération de test *(gen)* est illustrée par la construction du testeur canonique. Par définition et par construction du testeur canonique associé à *conf*, les propriétés de *correction* et *d'exhaustivité* sont satisfaites. Mais cela est théorique car, il faut en général exécuter des tests infinis [83].

3.2.5 MOOT : Méthodes orientées objectif de test

A la différence des deux précédentes classes de méthodes, les $MOOT$ ne visent pas à comparer le comportement global de l'IST avec celui de la spécification. Il s'agit de définir des propriétés, appelées *objectif de test* (OT), qui sont exprimés par un séquencement d'évènements (ou d'états). A partir d'un OT donné, un cas de test est généré. Ce cas de test décrit un séquencement d'évènements permettant d'observer la propriété définie par l'OT. La *conformité* s'exprime à l'aide d'un ensemble de *critères* définis par l'OT. Il faut rappeler que cette approche de la conformité était antérieurement pratiquée en milieu industriel (par exemple chez Alcatel), mais de manière empirique, avec une génération manuelle de tests. Des travaux récents proposent des démarches de génération automatiques [33, 20, 68, 27, 81].

Expression de critères de conformité par les OT

Un objectif de test n'est rien d'autre qu'une propriété, exprimable par une séquence d'évènements, que l'on voudrait observer dans le fonctionnement de l'IST. Les formalismes suivants peuvent être utilisés pour l'expression des OT :

- Expression rationnelle : dans la théorie des automates, une expression rationnelle est une formule qui définit un ensemble de séquences d'actions ou mots [2]. Une telle expression est construite à l'aide d'opérateurs tels que la concaténation ., le choix + et l'itération $*$. Un langage d'expression d'OT, basé sur les expressions rationnelles, a été proposé [68].

- Automates d'états finis : un OT peut être exprimé en terme d'automates [61, 27]. L'automate exprime alors le séquencement des interactions que l'on voudrait observer pendant le test.
- Diagrammes de séquences de messages DSM : un DSM [39] est un diagramme décrivant un enchaînement temporel d'interactions. Les flèches d'un DSM expriment des échanges de messages entre processus (ou processus et environnement).
- Chemin dans un graphe syntaxique : dans le contexte des TDF -(Technique de Description Formelle), on peut également définir des OT, en utilisant toujours le concept de séquencement d'actions. Cela est possible en extrayant de la spécification un modèle d'automate, ou en utilisant directement le graphe syntaxique associé à cette spécification [20, 81]. L'OT est alors un chemin dans ce graphe syntaxique. Ce chemin peut être décrit dans un langage proche de celui de la TDF utilisée.

Génération de cas de tests

Démarche générale : l'OT étant défini, il faut ensuite rechercher des cas de test satisfaisant cet OT. Intuitivement, la démarche consiste donc à parcourir en parallèle la spécification et l'OT, jusqu'à trouver des séquences adéquates. Cette démarche est pratiquée aussi bien pour des systèmes décrits avec des automates que pour des systèmes décrits avec des TDF. Dans ce dernier cas, les séquences recherchées peuvent être obtenues par simulation conjointe de la spécification et de l'OT [20, 81]. Les traces obtenues par simulation sont bien des séquences d'action vérifiant l'OT.
L'OT définit en fait des comportements particuliers de la spécification. En conséquence, en le synchronisant avec celle-ci, l'ensemble des comportements acceptés par le produit synchronisé résultant est nécessairement plus petit que celui de la spécification considérée séparément. Cependant pour certains systèmes, le nombre de séquences de test obtenues peut encore exploser. Il faudrait donc utiliser des techniques de recherche plus adaptées aux problèmes d'explosion.

Applicabilité, lien avec la norme

Il est clair que les $MOOT$ (et les heuristiques utilisées) sont pratiques, car elles permettent de générer des tests suivant un OT donné. Cela est intéressant dans la mesure où l'on peut produire des tests selon des besoins spécifiques. Cependant cette démarche n'est pas suffisante pour établir de manière claire une comparaison entre une spécification et une implantation données [61]. Il faudrait disposer d'un ensemble d'OT permettant de couvrir une définition globale de conformité. En effet un OT donné ne vise pas, en

général, à tester tous les comportements possibles.
Lien avec la norme : l'expression de la *conformité* s'apparente plutôt à l'utilisation de *REQS* (ensemble des *critères de conformité*), en définissant des propriétés particulières traduites en termes d'*OT*.
La propriété de *correction* est satisfaite par les MOOT [61, 27] mais, l'exhaustivité n'est pas garantie. Il est nécessaire de disposer d'un ensemble représentatif d'OT qui couvre de manière exhaustive une définition de conformité.
Pour la plupart des outils de génération automatique de test tels que TGV [22], TTCGEN, SAMSTAG [33], TVEDA-3 [20], le problème demeure l'explosion combinatoire. Ce qui limitait souvent l'applicabilité de la génération automatique à des spécifications de petite taille. C'est pour remédier à ce problème qu'il faudrait utiliser des techniques de recherche plus adaptées aux problèmes d'explosion telles que la recherche à la volée.

Description d'une recherche à la volée

La recherche de séquences est un problème d'accessibilité. Il s'agit de faire un parcours d'arbre d'accessibilité jusqu'à trouver les séquences recherchées (correspondant à l'OT). Plusieurs algorithmes existent (recherche en profondeur, recherche en largeur). L'heuristique choisie doit tenter d'éviter l'explosion combinatoire des chemins mémorisés pendant cette recherche. Une recherche en largeur explose donc puisqu'elle nécessite le stockage de tous les chemins intermédiaires rencontrés depuis la racine. Une recherche à la volée, basée sur un parcours en profondeur, convient [48]. Cependant, la longueur des séquences recherchées est bornée à une valeur k fixée. Pendant la recherche, si un chemin donné, de longueur k, ne vérifie pas l'OT, ce chemin est abandonné et un autre est exploré. La borne k peut être augmentée pour relancer la recherche.

3.3 Le test d'interopérabilité

3.3.1 Présentation

l'objectif du test de conformité est de tester la conformité d'une implantation par rapport à sa spécification, c'est-à-dire vérifier que l'implantation réalise bien les fonctions décrites dans la spécification. Le test de conformité est donc un test qui met en jeu une spécification et une implantation. Le test d'interopérabilité étudie quant à lui la capacité qu'ont deux ou plusieurs implantations à coopérer correctement sur un réseau afin de fournir le service attendu.

S'il est possible dans le cadre du test de conformité de vérifier que deux implantations sont conformes à une même spécification, cela ne garantit pas que ces deux implantations pourront interagir entre elles. Il est en effet pos-

sible que la norme laisse aux concepteurs la responsabilité de certains choix d'implantation critique pour l'interopérabilité.

Le test de conformité est une étape nécessaire pour assurer l'interopérabilité. Cependant, ce test n'implique pas forcément l'interopérabilité et vice versa car :

- le test de conformité ne peut être exhaustif, ceci en raison des limitations matérielles. Ainsi certains cas non testés peuvent être à l'origine de non fonctionnement.
- les normes peuvent être à l'origine de mauvaises interprétations. En effet celles-ci ne sont pas toujours rigoureusement décrites, notamment par des spécifications formelles. Certains problèmes de non interopération sont directement liés à cette limitation.
- les normes des protocoles permettent des options et choix qui peuvent être quelquefois contradictoires. Aussi des implantations, qui implantent des options ou choix différents, ne peuvent pas interopérer. Néanmoins pour pallier à ce problème, des profils fonctionnels associés aux implantations permettent de résoudre en partie ce problème. Ainsi une étude des profils doit limiter la non interopération. Mais ces profils comportent également quelques fonctions optionnelles. De plus l'étude des profils n'est pas une activité de test, mais une activité de validation de la norme par rapport aux besoins définis.
- le test de conformité ne fournit pas de tests pour les performances et pour la robustesse. Ainsi des implantations peuvent être conformes aux normes, mais en pratique elles peuvent avoir des performances insuffisantes pour interfonctionner efficacement.

3.3.2 Architecture

Selon la façon dont les implémentations à tester sont interconnectées et selon le degré avec lequel nous pouvons observer leur interaction, différentes architectures de test d'interopérabilité peuvent être utilisées. Plusieurs architectures ont été proposées dans la littérature [12, 69, 3, 86]. Considérons l'architecture générale [4] de la figure 3.3 dans lequel le SST (système sous test est composé de deux ISTs (implantation sous test IST1 et IST2). Chaque IST est une boîte noire. L'interface supérieure IS_i (resp. l'interface inférieure II_i) est l'interface de l'ISTi, à travers lequel il communique avec ses couches supérieures (resp. inférieures).

Les services attendus sont fournis à travers l'IS_i pendant que l'II_i est utilisée par l'ISTi pour interagir avec l'entité homologue de la même couche. Différents contrôles et observations sont possibles en fonction de l'environnement de test, ainsi le testeur supérieur TS_i (resp. testeur inférieure TI_i) est la partie du système testeur en charge des contrôle et/ou des observations de l'IS_i (respectivement II_i), à travers le PCO supérieure $PCO - Sup_i$ (resp.

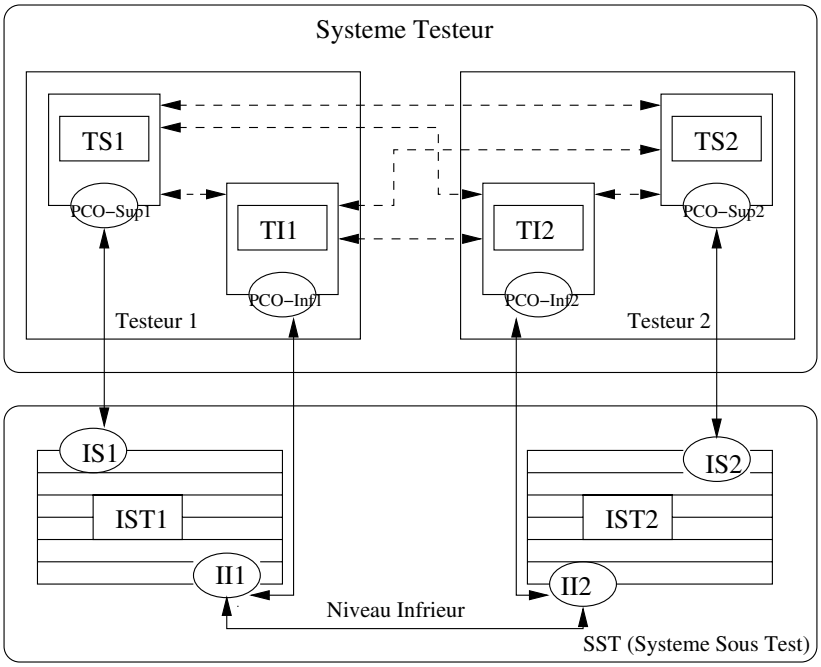


FIG. 3.3 – Architecture générale du test d'interopérabilité

PCO inférieure $PCO - Inf_i$). Le testeur T_i (composé de TS_i et/ou TI_i) est la partie du ST en charge des contrôles et observations des ISTi.

Pour tester l'interopérabilité, différentes architectures ont été définies. Ces architectures considèrent, comme pour le test de conformité, deux types de test : le test "boîte noire" et le test "boîte grise".

Pour le test "boite noire", les échanges ne peuvent être observés qu'à travers les interfaces externes du système, les points de contrôle et d'observations (PCOs), grâce auxquels il interagit avec son environnement. Le test d'interopérabilité "boîte noire", ou test actif, ne permet pas d'observer la communication entre les deux entités sous test. Il n'y a pas de PCO au niveau des interfaces inférieures des entités. Le test d'interopérabilité "boîte grise", quant à lui, permet en plus des PCOs aux interfaces supérieures, une observation de la communication entre les entités. Avec ces approches, les cas de non interopérabilité sont analysés en aval, afin d'en déduire les raisons.

Quelques considérations

– *Boîte noire contre boîte grise* : c'est bien connu que le test de conformité est effectué dans un contexte de boîte noire. Mais il y a encore

un débat dans le test d'interopérabilité entre le contexte boîte grise et boîte noire. Le point de contestation vient de la possibilité d'observer ou non (et/ou contrôler) les interactions entre les implémentations qui composent l'IST (implantation Sous Test).

– *Protocole contre service* : la définition de l'interopérabilité peut être "orientée couche" [3, 31]. Par exemple dans [3], quatre niveaux d'architecture d'interopérabilité appelé respectivement *protocole*, *service*, *application* et *utilisateur* sont considérés.
– *Active contre passive* : si le système de testeur a la possibilité de stimuler et/ou corrompre les évènements dans le SST (système sous test), les architectures de test correspondant sont appelées actives [69], sinon elles sont appelées passives [86, 31].
– *moniteur contre arbitre* : dans ces architectures [86], un composant de test est placé entre les implémentations pour contrôler (moniteur) ou observer (arbitre) leur communication. Etre moniteur ou arbitre dépend du caractère actif/passif du comportement du testeur. Les arbitres sont utilisés pour identifier les implémentations défectueuses [24], pendant que les moniteurs sont utilisés pour avoir un meilleur contrôle du test.

3.3.3 Problématique de l'interopérabilité

La problématique d'interopérabilité mène vers deux aspects : la construction et la validation de l'interopérabilité.

– *construire l'interopérabilité*, c'est développer des normes de telle sorte que l'interopérabilité soit assurée. C'est la problématique de comment faire pour obtenir le meilleur degré d'interopération entre les implantations. Le principe est simple, il faut mettre le moins d'hétérogénéité possible ou faire en sorte qu'elle soit vue ainsi.
– *valider l'interopérabilité* d'un ensemble d'équipements, c'est évaluer jusqu'à quel degré ces équipements peuvent communiquer et coopérer. C'est la problématique du comment faire pour connaître le degré d'interopérabilité. Dans les premiers temps, les organisations travaillant sur les protocoles de communication ont cru que cette propriété serait satisfaite dès lors que tous les équipements seraient conformes à leurs spécifications. Le test de conformité peut préconiser jusqu'à quel degré de confiance un équipement respecte la spécification dans son comportement. Mais, du fait qu'une norme peut souvent être sujette à des interprétations au niveau des options, des paramètres, etc.. il arrive que des implantations d'une même spécification des constructeurs différents présentent des caractéristiques différentes, bien qu'elles soient toutes conformes. Benkhellat dans [6] définit deux types de validation de l'interopérabilité :
  • validation par test (expérimentale) qui consiste à appliquer des sé-

quences de test à un système sur une plate-forme ;
- validation par vérification (théorique, nommée aussi à priori) qui consiste à comparer dans un modèle de valeurs certaines caractéristiques pertinentes pour l'interopérabilité.

L'activité de validation de l'interopérabilité par le test reste au stade d'expérimentation concrète et est encore loin d'une quelconque normalisation. mais elle identifie neanmoins plusieurs phénomènes et intègre différents aspects qui ne sont pas couverts par le test de conformité [5] :

– interprétation des normes dans une implantation donnée qui ne serait pas détectable dans un test de conformité.
– nécessité de prendre en compte le cas ou plusieurs implantations coopèrent ensemble. Par conséquent, il y a besoin de considérer le test de connexions simultanées multiples.
– considération des ressources à mettre en oeuvre et des performances temporelles requises des ISTs (Implantations sous test).
– hétérogénéité, contrairement à la conformité qui compare une entité à un testeur sensé respecter la spécification, l'interopérabilité confronte des implantations hétérogènes.
– l'aspect d'une prise en compte des phénomènes des environnements réels tels que l'interconnexion de plusieurs implantations, la communication d'une implantation avec plusieurs autres implantations à la fois, et le fait que les échanges sont très variés.

Nos travaux s'inscrivent dans l'aspect vérification, validation de l'interopérabilité par test et expérimentation. Nous proposons par suite (voir section 3.8) une stratégie de test de l'interopérabilité. L'expérimentation consiste à observer l'exécution des protocoles et services dans un environnement réel tout en permettant le contrôle et l'observation de l'ensemble des comportements possibles de la spécification.

3.3.4 Contexte de recherche

Les auteurs de [70, 13] ont été des précurseurs dans le domaine du test d'interopérabilité. Le test d'interopérabilité est réalisé à travers les interfaces utilisateurs avec chaque système sous test. La génération de test d'interopérabilité nécessite la description de la communication globale entre les deux entités sous test, primitives de services comprises. Ils définissent également les architectures de test et l'approche de *test à cheval*. Cette méthode, appelée *"méthode à cheval"*, permet de résoudre les problèmes de synchronisation de testeurs qui existent dans les méthodes de test distribuées.

O. Koné dans [12] définit formellement la relation d'interopérabilité et son testeur canonique. Il reprend la théorie du test définie pour la confor-

mité et notamment la relation d'implantation pour l'adapter à une relation d'implantation d'interopérabilité. Il propose une méthode de génération au vol des séquences de test. Le système de test qu'il propose est un testeur canonique d'interopérabilité. Le problème avec cette méthode est que les tests générés sont longs.

Y. Benkhellat [6] présente en 1995 une très bonne analyse du problème de l'interopérabilité. Les services , les protocoles, les ressources et les performances temporelles sont considérées comme étant les quatre grandes classes caractéristiques de conformité des équipements influant sur leur interopérabilité.

G. Leduc [50, 49] a proposé la relation d'implantation *imp* pour lier une implantation I à sa spécification S : I *imp* S. Comme la relation *imp* n'est pas toujours transitive (elle est une composition des raffinements intermédiaires faits à partir de S jusqu'à l'obtention de I), l'auteur a proposé la relation d'implantation restreinte, *imp_rest* pour lier les spécifications intermédiaires ou les raffinements de la spécification S. La relation *imp_rest* exprime le fait de raffiner une spécification ne peut que réduire les incertitudes concernant son implantation.

Ainsi, deux implantations d'une spécification ont une plus grande probabilité d'interopérer quand elles partagent plus de raffinements intermédiaires de ladite spécification.
Il convient de noter que les travaux de G. Leduc ne traitent pas directement de l'interopérabilité des systèmes, son apport sur l'aptitude à interopérer d'un ensemble d'équipements dérive de son étude sur les relations d'implantation.

Aux Bell Laboratories [34], on utilise la notion d'interopérabilité pour caractériser la possibilité d'obtenir les services demandés de la part des systèmes en communication et qui doivent effectuer ensemble une activité spécifique. Les systèmes peuvent être locaux ou distribués, hétérogènes ou non. Les causes d'une éventuelle non-interopérabilité sont générées soit parce que les spécifications décrites par les normes sont ambiguës, soit parce qu'il existe des extensions introduites par les divers constructeurs. Cette étude se caractérise par une description efficace des structures de données et des algorithmes de génération de tests, qui garantissent une couverture de test désirée et qui produisent un nombre minimal de tests.

Un organisme, le ASD (Advanced Studies Department du COS - Corporation for Open System), a proposé les définitions suivantes dans [31] que [6] reprend dans sa thèse :

– *L'interopérabilité testable des couches* - une implantation d'un proto-

cole à une couche N est dite interopérable par test si elle a passé avec succès les tests de conformité du protocole et qu'il a été montré qu'elle réalise avec succès la fonctionnalité du service N avec une sélection de couches homologues (par exemple les N constructeurs les plus importants) comme elle a été spécifiée par le standard et exigée par le profil quand le service (N-1) spécifié et offert est utilisé. On désigne dans la littérature le test de l'interopérabilité des couches sous le terme de test horizontal.

– *L'interopérabilité testable de système* - Un système informatique (ou station) est dit système interopérable par test si chaque implantation des protocoles qui font partie du système est une couche interopérable par test et qu'il a communiqué avec succès avec une sélection d'autres systèmes.

3.3.5 Les problèmes relatifs au test d'interopérabilité

L'approche que nous utilisons pour la génération des séquences de test d'interopérabilité consiste dans l'application de techniques correspondantes au test de conformité. Pour la génération systématique de suites de test d'interopérabilité, les problèmes les plus difficiles [37] à résoudre sont :

– La conception de l'abstraction formelle relative au comportement du système.
– Etablir les hypothèses de test afin d'obtenir une couverture satisfaisante de test.
– L'explosion combinatoire d'états des automates caractérisant les comportements globaux des systèmes, déterminée par la simultanéité des évènements et par la complexité des comportements des composants [12, 45].
– L'obtention des séquences de test garantissant une couverture satisfaisante des fautes.
– L'automatisation de la génération de séquences de test.
– L'application des séquences de test obtenues sur une plate-forme de test.

La conception de l'abstraction formelle, caractérisant le comportement du système, est fortement dépendante d'hypothèses de test établies. Ces hypothèses doivent être choisies afin d'assurer une couverture de test satisfaisante et pertinente [66, 18]. Les experts établissent les hypothèses de test sur la connaissance qu'ils ont du système et sur leur expérience du test.

3.3.6 Hypothèses relatives au test d'interopérabilité

Il est admis que la qualité d'un test s'exprime par sa couverture, c'est à dire une évaluation (éventuellement chiffrée) devant rendre compte de l'aptitude du test à détecter les fautes et dans le cas échéant, l'implantation est

dite correcte. Les contraintes de coût et de temps ou tout simplement l'impossibilité technique de générer des tests exhaustifs nous obligent à faire une sélection rigoureuse de quelques scénarios de test. Ce choix est souvent guidé par la connaissance des équipements et s'appuie sur des hypothèses simplificatrices. Ces hypothèses représentent aussi une façon de gérer le problème d'explosion combinatoire relatif au graphe d'accessibilité engendré à partir d'une spécification formelle d'un protocole ou d'un service.
Faire une hypothèse de test consiste à supposer que les implantations testées, qui sont en réalité des objets physiques, peuvent être correctement modélisées par un ensemble d'objets mathématiques donné.

Une hypothèse A est plus forte qu'une hypothèse B si l'ensemble des modèles des implantations selon l'hypothèse A est inclus dans l'hypothèse B [66]. Ainsi, on caractérise la conformité, l'interopérabilité ou l'interfonctionnement sous réserve que l'implantation testée satisfasse l'hypothèse de test choisie. la qualité d'exploration de la spécification dépend des poids d'hypothèses choisies [17], il y a donc un compromis à faire entre la couverture de test et "l'exhaustivité". Excepté pour quelques travaux triviaux, aucune méthode de couverture proposée jusqu'à présent ne permet de prouver qu'un test garantit le bon fonctionnement d'un système.

- *L'hypothèse d'uniformité* permet de considérer qu'une transition étendue est valide si on n'a testé que quelques uns des messages étiquetant la transition. Ceci est fréquemment utilisé pour des messages contenant des paramètres. Un exemple présenté par Phalippou est le suivant : considérons une unité de données de protocole : DATA(n : entier). Le paramètre n implique qu'il existe un grand nombre d'instances de cette unité de données. L'hypothèse d'uniformité considère que toutes les instances sont équivalentes du point de vue du test et qu'il suffit de tester une seule. Cette hypothèse permet donc de réduire un ensemble de transitions uniformes à une seule. C'est le cas lorsqu'un message d'entrée est paramétré, l'idée est de se dispenser à prendre toutes les instances de message et d'en fixer une.

- *L'hypothèse d'indépendance* considère que si la machine d'états finis est le résultat du produit indépendant de deux machines d'états finis plus simples, alors le test de ces deux dernières indépendantes est équivalent au test de la composée. Cette hypothèse considère aussi que si la spécification prévoit qu'un état d'une machine à états finis peut être atteint par plusieurs chemins, alors dans l'implantation, cet état est indépendant du chemin qui y conduit.

- *L'hypothèse d'équité* permet de réduire l'indéterminisme de certains systèmes. Cette hypothèse permet de dire qu'au bout d'un nombre d'essais raisonnable, on aura fait tous les choix permis par l'implanta-

tion.

3.4 Stratégie de test

3.4.1 Evaluation de besoins en matière de test

Nécessité de formaliser

Un des problèmes qui se posent lors de la normalisation d'un protocole et/ou d'un service, et même plus généralement lors de leur conception, est le choix du formalisme utilisé pour rédiger les spécifications. Les normes WAP ont été décrites en langage informel (naturel). Cela manque de précision, ce qui peut entraîner des problèmes dans la conception et dans la compréhension du protocole. Le langage naturel autorise la rédaction des spécifications ambiguës. Il y a par suite interprétation du texte par l'être humain qui le lit. Or qui dit interprétation dit plusieurs interprétations possibles. Ce qui conduira inéluctablement à développer des implantations incompatibles. Ceci peut introduire de graves problèmes d'interopérabilité entre les systèmes en communication. Par ailleurs, la génération automatique de test nécessite une spécification formelle.

Le problème d'explosion combinatoire

Pour effectuer un "bon" test, il est nécessaire de maîtriser le problème de l'explosion combinatoire. Plusieurs approches ont été développées afin d'éviter ce problème qui apparaît pendant l'analyse du modèle du système, lors de la construction du graphe d'accessibilité en utilisant des méthodes classiques. Ces méthodes génèrent le graphe d'accessibilité en décrivant le comportement global complet du système étudié :

- Une première solution pour résoudre le problème de l'explosion combinatoire consiste dans l'application d'une approche d'ordre partielle, permettant de ne pas mémoriser tous les chemins dans le graphe généré par des transitions entrelacées.
- Une deuxième solution consiste dans l'approche "on the fly" [42, 41, 43], qui consiste à construire le graphe du système étudié de façon paresseuse (selon le besoin), par l'exploration du graphe.
- La troisième solution consiste dans l'approche par minimisation, basée sur des abstractions de spécifications. Ces abstractions peuvent être formulées en terme de transformations classiques d'équivalences sur des automates.

Dans nos travaux de recherche, grâce à l'outil TestComposer [46] (intégré à la version actuelle de l'outil ObjectGEODE) de Telelogic,nous appliquons une méthode orientée objectif de test en vue de limiter l'explosion combinatoire. La démarche consiste à parcourir en parallèle l'OT et la spécification

jusqu'à l'obtention d'une séquence adéquate en utilisant la technique de simulation et de recherche à la volée implémenté dans dans cet outil.

3.4.2 Méthodologie et architecture de test

Méthodologie de test

Notre objectif est de définir une méthodologie et des architectures pour la validation et l'expérimentation de services et protocoles. Il s'agit également de couvrir de manière automatisée toutes les étapes de la production des tests : de la spécification du système à tester à leur exécution sur la plateforme réelle. Cette dernière permet de traiter deux types de test : interopérabilité et conformité. Le test d'interopérabilité permet de valider que des implantations fournissent le service global attendu en étant conforme aux standards. Il permet par exemple de vérifier l'interopérabilité entre une application sur un terminal et une application sur un serveur. Le test de conformité vérifie que l'implantation sous test est conforme à sa norme (par exemple, une couche protocolaire WAP).

Notre méthodologie de test est basée sur deux éléments principaux : les méthodes de génération et les architectures.

Nous décrivons dans ce qui suit les étapes de notre méthodologie :

1. La première étape de notre méthodologie consiste en l'obtention d'une représentation précise du système à étudier. Le langage de description que nous utilisons est le langage SDL [40]. La spécification doit tenir compte de l'architecture de test et de ses interactions avec l'environnement .
2. La deuxième étape consiste en la sélection des tests pertinents. Il s'agit de choisir les tests à effectuer, en se basant sur divers critères (couverture, modèles de fautes, hypothèses de test, coût acceptable de test, etc..). Cela correspond dans la méthodologie ISO 9646 à la définition des buts de tests.
3. Dans la troisième étape, nous calculons les tests : étant donné un test défini sous forme de but de test, il s'agit de calculer la séquence d'interactions qui, appliquée à l'implantation sous test, va vérifier un tel but. Dans cette étape, une méthode de génération de tests doit être définie afin de calculer les tests. La méthode de génération de test prend en compte le type de système sous test, selon qu'il est constitué d'un seul module ou de plusieurs modules communicants.
4. Dans la quatrième étape, nous devrons mettre en forme les séquences de test générées. Il s'agit de la production de suites de tests exploitables dans un formalisme reconnu et adapté. Dans le domaine de systèmes de télécommunications, TTCN [1] s'impose peu à peu comme une solution unanimement retenue. Les séquences de test peuvent aussi être

transformées dans un format adéquat pour le test réel. Cette transformation fait référence à l'idée de codage binaire de messages, en utilisant par exemple la notation ASN.1 [25].

5. Enfin, les tests ainsi générés sont exécutés sur une plate-forme réelle muni des POs et des PCOs qui nous permettent d'observer et de contrôler les différents échanges en vue de vérifier la conformité des différentes entités du système par rapport aux normes de référence et de valider l'interopérabilité du système global.

Pour atteindre les objectifs ci-dessus, nous avons proposé les solutions suivantes qui seront détaillées dans le chapitre 4 :

- Dans la première étape, nous avons utilisé le langage SDL afin de décrire formellement le comportement du système WAP. Plus particulièrement, nous avons décrit en SDL les comportements de protocoles WSP et WTP, et les services de localisation dans les réseaux UMTS.
- Dans la deuxième étape, la sélection des tests prend en compte l'établissement d'une session, l'invocation d'une méthode, la reprise de la session, la déconnexion. Il convient de préciser ici que nous aurons pu choisir d'autres objectfs et la méthode serait la même. Nous pensons simplement que l'établissement d'une session et l'invocation de méthodes par exemple sont des opérations nécessaires lors de l'échange de données.
- Dans la troisième étape, nous proposons l'utilisation de la méthode automatique d'exploration partielle des graphes et de l'outil de génération de test TestComposer [46]. La méthode de génération de tests utilisée par l'outil Test Composer permet d'obtenir une couverture satisfaisante par rapport aux suites de tests sélectionnés dans la deuxième étape de la validation. Test Composer est inclut dans l'environnement ObjectGEODE [79].
- Dans la quatrième étape, nous produisons des suites de test dans le formalisme MSC [39] ou TTCN [1].
- dans la cinquième étape nous exécutons les tests sur la plate-forme mise en oeuvre dans le cadre de nos travaux de recherche.

Architectures de test

La plate-forme a été appliquée dans un premier temps pour valider et expérimenter des services liés aux réseaux mobiles tels que le WAP sur GSM/GPRS et l'UMTS. Du fait de l'hétérogénéité et de la complexité des éléments du réseau, les architectures classiques de test ne peuvent convenir [71], car plusieurs entités doivent coopérer à travers le réseau pour fournir le service désiré. C'est pourquoi, nous proposons l'architecture de test suivante (fig.3.4) :

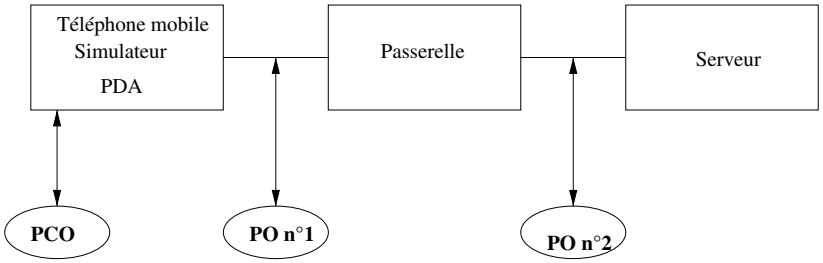


FIG. 3.4 – Architecture générique pour le test d'une application mobile

Pour pouvoir analyser les échanges de données, il est nécessaire d'observer le transit en différents points stratégiques. Ceci conduit à l'installation de Points d'Observation (POs). Un PO (PO n°1) est donc installé entre le terminal mobile et la passerelle afin de pouvoir collecter les données contenues dans la liaison hertzienne et ainsi vérifier l'interopérabilité entre ces équipements. Le deuxième PO (PO n°2), entre la passerelle et le serveur permet d'observer si les données sont bien retranscrites lors du transfert de la passerelle au serveur. Le Point de Contrôle et d'Observation (PCO) situé à l'amont de l'appareil mobile permet d'initialiser des transactions, d'injecter des événements (valides, inopportuns ou invalides) et de récupérer les résultats.

Avec cette architecture, le pouvoir de détection des erreurs (capacité à détecter les erreurs) est assez fort. En effet, nous capturons toutes les données échangées qui servent à réaliser un service donné. Ainsi, si une erreur survient, entre la passerelle et le serveur, c'est le PO situé sur cette connexion qui détectera le problème en premier. Il n'est pas obligatoire de conserver tous ces POs, mais pour chaque PO enlevé, le pouvoir d'observation diminue. Une possibilité pour optimiser au maximum les scénarios de tests consiste à les faire sans aucun PO, et si les tests échouent, rajouter les POs afin de pouvoir diagnostiquer plus facilement où l'erreur (ou les erreurs) s'est produite. Ceci introduit une méthodologie incrémentale de test.

Verdict obtenu

Afin de vérifier que le test se déroule normalement, chaque interface est couplée avec un PO, ce qui nous conduit à en installer un par zone. (Pour la figure 3.4, nous n'avons que deux interfaces : une entre le terminal et la passerelle, et l'autre entre la passerelle et le serveur). A partir de chaque PO un corpus de traces est sauvegardé. Des propriétés liées aux objectifs de test seront vérifiées sur les traces afin de détecter et localiser les erreurs. Des verdicts locaux sont produits. Par ailleurs, chaque PCO dispose donc d'un scénario de test par rapport auquel la conformité est vérifiée pour fournir un verdict local à la zone. L'ensemble des verdicts locaux est

retourné à un testeur central dont le rôle est de déduire le verdict final *FAIL*, *INCONCLUSIVE* ou *PASS*. Cette architecture peut être vu comme distribuée.

Configuration de la plate-forme

La plate-forme dispose de différents sites correspondant aux partenaires du projet (figure 3.5). L'accès à la plate-forme peut être fait soit en utilisant un téléphone mobile, soit un PDA, soit un simulateur de terminal mobile. Pour les téléphones mobiles et les PDA, l'accès par authentification au site se fait en utilisant un RAS (Remote Access Service). Ce dernier nous permet d'accéder à la pile protocolaire WAP Kannel, qui est un logiciel Open Source. Celle-ci connecte les terminaux mobiles aux différents serveurs HTTP et/ou WAP existants sur notre plate-forme.

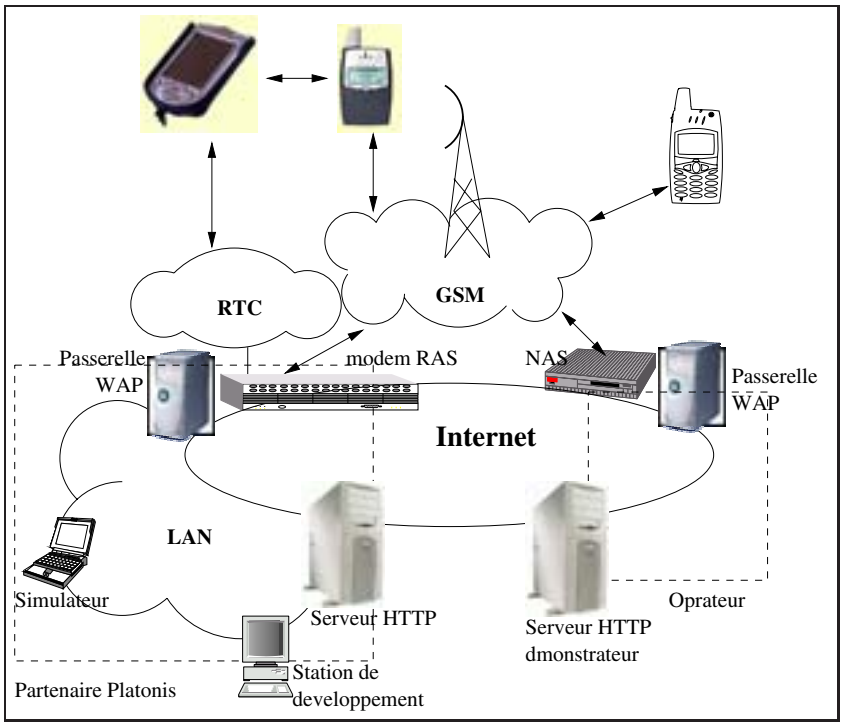


FIG. 3.5 – Plate-forme PLATONIS

3.5 Conclusion

Ce chapitre fournit une synthèse des travaux développés dans le cadre du test de conformité et d'interopérabilité d'une part et propose la stratégie

de test du système WAP et UMTS. Les méthodes relatives au test d'interopérabilité sont classifiées en trois grandes approches : les MAEF, les MTTC et les MOOT. Chaque approche s'appuie sur des modèles formels qui sont des systèmes de transitions. Les MTTC ont contribué à formaliser et à comprendre le processus de test. Mais ces méthodes n'ont pas été réellement appliquées à des cas complexes et elles n'ont pas donné lieu à de véritables outils.
Les MAEF et les MTTC visent à établir une *relation de conformité* entre le comportement global d'une IST et celui d'une spécification. Ces méthodes permettent d'obtenir des tests exhaustifs mais sont confrontées au problème d'explosion combinatoire. Pour les systèmes de taille importante, le graphe décrivant l'ensemble des comportements est généralement de taille très élevée. Manipuler un tel graphe en vue de calculer des suites de tests pose des problèmes de stockage en mémoire. Les MOOT proposent une alternative au problème d'explosion à l'aide de méthodes qui génèrent les tests en fonction de propriétés spécifiques (*critères de conformité, OT*). Les techniques de génération de tests à la volée et les techniques de simulation permettent de sélectionner des tests, en tentant d'éviter les problèmes d'explosion combinatoire. S'il est possible dans le cadre du test de conformité de vérifier que deux ou plusieurs implantations sont conformes à une même spécification, cela ne garantit pas que ces implantations pourront interagir entre elles. Le test d'interopérabilité (ou d'interfonctionnement) met en relation des implémentations. Il s'agit alors de s'assurer que ces implémentations interagissent conformément à ce qui est décrit dans leur(s) spécifications(s). Nous avons dans ce chapitre, proposé une stratégie de test basée sur l'utilisation des méthodes formelles et la validation expérimentale.

Chapitre 4

Spécification des services et protocoles, génération de tests

4.1 Introduction

Le travail présenté dans cette thèse est motivé par la validation des systèmes de communication décrits en SDL.
Ce chapitre poursuit l'étude et le détail de la stratégie de test utilisée dans nos recherches. Les normes WAP [28, 29] ont été décrites en langage informel (naturel). Cela manque de précision, ce qui peut entraîner des problèmes dans la conception et dans la compréhension du protocole. De même, le langage naturel autorise la rédaction des spécifications ambiguës. Ce qui conduira ensuite à développer des implantations incompatibles incapables de communiquer entre elles de la façon attendue. Ceci peut introduire de graves problèmes d'interopérabilité entre les systèmes en communication. D'autre part, l'absence de spécification formelle met un frein à la génération automatique de test. Pour résoudre ces problèmes, nous avons fait le choix d'utilisation du langage SDL[1]. Dans ce chapitre, nous détaillons la technique que nous avons proposée dans la section 4 du chapitre 3 selon le plan ci-dessous.

La *deuxième section* de ce chapitre présente d'abord les différentes techniques de description formelle et en particulier le formalisme SDL utilisé dans le cadre de ce travail de recherche. Ensuite, est présenté l'environnement ObjectGEODE qui permet d'éditer, de simuler et de générer des tests pour nos services et protocoles.

[1]Specification and Description Language

La *troisième section* du chapitre propose une méthode de génération de tests pour la couche applicative du WAP (la couche WAE). En effet, dans l'environnement WAP, la couche WAE fournit un cadre de spécification d'applications qui sont définies en langage WML[2]. Ces services correspondent à l'exécution de programmes interagissant directement ou indirectement avec les utilisateurs finaux munis de leur mobile. Les tests de ce service doivent permettre de vérifier qu'une application est bien écrite et répond aux attentes des utilisateurs finaux. Dans cette section, nous avons donc proposé une méthode pour valider une application écrite en WML et par conséquent la couche applicative WAE.

Dans la *quatrième section* de ce chapitre, nous formalisons et générons des tests pour les services fournis par les couches protocolaires, ces services sont spécifiés dans les normes. Nous nous sommes intéressés aux couches protocolaires WSP[3] et WTP[4]. Ces tests nous permettent de valider la conformité, l'intéropérabilité locale "verticale" et intéropérabilité distante "horizontale" (modélisation et génération de séquences entre entités protocolaires de même niveau).

Par suite, la *cinquième section* présente le service de localisation (LCS service). Ce service met en oeuvre des mécanismes qui permettent au serveur de localisation (LCS-serveur) de calculer les positions (information de localisation ou LCS-information) des mobiles afin de les fournir au serveur (LCS-client) sur demande autorisée. Le serveur pourra ainsi renvoyer au terminal des informations personnalisées qui tiennent compte de sa position. Le problème qui se pose ici est donc celui de fournir avec fiabilité la position du terminal. En effet, plusieurs entités hétérogènes du LCS-serveur interopèrent, un test d'interfonctionnement va donc permettre de vérifier la fiabilité de l'opération qui consiste à fournir l'information de localisation.

Plusieurs travaux ont été publiés sur l'analyse de performance [35, 64] concernant les services de localisation, mais nous n'avons pas connaissance des études concernant la simulation fonctionnelle et la conception de tests basés sur des techniques formelles dans les réseaux UMTS notamment. Ce travail constitue donc une originalité concernant les LCS services et même la méthodologie de test d'une application WML. Une autre contribution réside dans la détection de certaines erreurs dans les protocoles qui peuvent nuire à l'interopérabilité du système global.

[2]Markup Wireless Language
[3]Wireless Session Protocol
[4]Wireless Transport Protocol

4.2 Quelques concepts

4.2.1 De la norme à la modélisation formelle

Le travail des comités de normalisation consiste à éditer des descriptions (normes ou standards) des services et protocoles, constituant la référence à laquelle toute implémentation doit se conformer.
Lorsque l'ISO[5] a commencé à normaliser les protocoles de communication, les spécifications produites étaient écrites en langage naturel.
L'application des techniques formelles a été envisagée afin de permettre la spécification des protocoles en réduisant au minimum le manque d'adéquation du langage naturel pour exprimer le type de contraintes liées à l'interconnexion des systèmes ouverts. Un des problèmes qui se posent lors de la normalisation d'un protocole et/ou d'un service, et même plus généralement, lors de leur conception, est le choix du formalisme utilisé pour rédiger les spécifications.

4.2.2 Les techniques de description formelle

La sous-section précédente donne des arguments en faveur des méthodes formelles de test. Les TDF[6] ont été conçues pour spécifier les systèmes de communication. Elles sont une solution au manque d'adéquation du langage naturel pour exprimer le type de contraintes liées à l'interconnexion des systèmes ouverts. Parmi les langages de description utilisés dans les TDF on trouve :

- Les modèles de base : FSM[7], etc..
- Les langages normalisés : Estelle [77], Lotos [26], SDL [40], etc..

Comme nous l'avons mentionné ci dessus, nous avons utilisé dans le cadre de ce travail le formalisme SDL qui est aujourd'hui très utilisé et apprécié dans le monde industriel pour la conception des systèmes de télécommunications.

4.2.3 Le formalisme SDL

Présentation du langage

SDL [40] est un langage normalisé par l'ITU. Il est largement utilisé dans l'industrie pour la description et spécification des protocoles de communication. Des implantations sont obtenues à partir de la spécification des systèmes en SDL. Ce langage est fondé sur le modèle des EFSMs [53] et utilise la communication asynchrone, par file d'attente. On dit que la communication est asynchrone si le processus peut transmettre quand il veut

[5]International Organization for Standardization
[6]Techniques de Description Formelle
[7]Finite State Machine

sans devoir attendre. Cette forme de communication est appelée communication par message. Elle exige une file d'attente non bornée afin d'éviter la perte des messages. SDL possède deux versions :

- SDL-GR : représentation graphique à l'aide des symboles normalisés ;
- SDL-PR : représentation textuelle, analogue à celle d'un langage de programmation.

Objectifs

Les objectifs du langage SDL sont de spécifier, décrire sans ambiguité des systèmes de télécommunication, de représenter les propriétés structurelles et fonctionnelles d'un système.
Les propriétés structurelles d'un système portent sur l'architecture du système et sa décomposition sous forme de blocs fonctionnels interconnectés. L'architecture d'un système SDL est organisée hiérarchiquement et fondée sur trois modèles sémantiques :

- *l'architecture du système*, qui contient la décomposition du système en termes d'entités fonctionnelles interdépendantes : système, blocs, sous-blocs, canaux, processus ;
- *le comportement d'un système*, donc l'ensemble des processus coopérants, communiquant par échange de signaux et variables partagées (le comportement dynamique de chaque machine, ses interactions avec les autres machines et l'environnement). Cela signifie une machine d'états fini étendue EFSM. Les propriétés comportementales d'un système caractérisent les réactions du système aux stimuli en provenance de son environnement. Les concepts objets sont introduits dans les versions normalisées SDL96 et SDL2000.
- *les données d'un système*

4.2.4 Le formalisme MSC

Une façon adéquate de décrire le comportement attendu d'un système est offerte par les traces du système qui sont convenablement présentées sous la forme des MSC[8] [39] car en plus d'une preuve générale de correction (comme l'absence de blocage), la cohérence de la spécification SDL avec le comportement souhaité du système doit être vérifiée.
Le MSC sont des moyens très répandus pour la description et particulièrement la visualisation graphique de traces de systèmes répartis, spécialement les systèmes de télécommunication. Un MSC montre les séquences des messages échangés entre les entités (tels que les services SDL, les processus, les blocs) et leur environnement. Formellement un MSC décrit un ordre partiel d'évènements, comme l'envoi et la consommation de message. Un MSC peut être déduit d'une spécification de système SDL existante. Cependant,

[8]Message Sequence Charts, ou Diagramme de Séquence de Messages (DSM)

un MSC est généralement créé avant la spécification du système et il constitue alors :

- une formalisation de besoins pour les spécifications SDL ;
- une base pour la génération automatique de squelette de spécifications SDL ;
- une base pour la sélection et la spécification de cas de test ;
- une spécification semi-formelle de la communication ; ou
- une spécification d'interface

Dans notre cas, les MSC seront surtout utilisés comme formalisme pour la sélection et la spécification des objectifs de test (ou propriétés à vérifiées) et les scénarios obtenus lors de simulations interactives.

4.2.5 L'environnement ObjectGEODE

Dans nos recherches, nous utilisons l'environnement ObjectGEODE [80], réalisé par la société Vérilog (Télélogic actuelle). Cet environnement offre une solution pour le développement des applications distribuées et temps réel. ObjectGEODE gère en parallèle les notations SDL [40] et MSC [39], pour l'édition, la simulation, la génération du code et le déboggage (voir figure 4.2).

- *l'éditeur ObjectGEODE SDL* : il gère le diagramme des arbres de blocs, le diagramme d'interconnexion, la partie de la machine à états finis et les déclarations de données ;

- *l'éditeur ObjectGEODE MSC* : il décrit à l'aide de diagrammes les scénarios d'échange d'informations entre les composants du système et l'environnement. Le MSC permet de valider la spécification décrite en SDL. Les descriptions SDL et MSC peuvent être consultées en interactif et les erreurs peuvent être corrigées rapidement. L'éditeur ObjectGEODE est capable de gérer des projets de grande taille.

- *le simulateur ObjectGEODE* : il teste le comportement du système pour vérifier si la spécification a été bien établie et pour obtenir des scénarios de référence qui peuvent être utilisés dans les ISTs. Pendant la simulation, peuvent apparaître des problèmes tels que les situations de blocage (*deadlock*, *livelock*), des pertes de signaux, un dépassement de la capacité des files d'attente, des erreurs dynamiques (les sorties illégales, les réponses inexistantes pour une décision, etc.). Le problème de *deadlock* fait que le système s'arrête définitivement. En revanche, dans le cas d'un *livelock*, le comportement est difficile à tester, parce que le système continue à fonctionner, mais de manière dégradée.
  Le but de la validation est de montrer que le modèle SDL accomplit correctement les services pour lesquels il a été conçu. Le simulateur

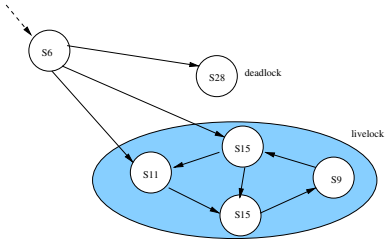


FIG. 4.1 – Deadlock et livelock

ObjectGEODE réalise la vérification et la validation des systèmes d'un point de vue qualitatif : livelock, deadlock, erreurs, etc.. Trois modes de simulation sont supportés par cet environnement :
- *le mode interactif* : l'utilisateur joue le rôle de l'environnement et observe le système comme une boîte noire ;
- *le mode aléatoire* : le simulateur exécute le modèle et fait un choix aléatoire quand se présente un situation à choix multiple ;
- *le mode exhaustif* : le simulateur exécute automatiquement le modèle et examine toutes les possibilités - exploration de tous les états du système et retour-arrière lorsqu'une branche a été totalement couverte.

- *Le générateur d'application ObjectGEODE* : il génère le système final. Le code C est obtenu en partant de la spécification SDL.

4.3 Méthodologie de test de la couche applicative WAE[9]

Cette section propose une méthode de génération de tests pour la couche applicative du WAP (la couche WAE). En effet, dans l'environnement WAP, la couche WAE fournit un cadre de spécification d'applications qui sont définies en langage WML[10]. Ces services correspondent à l'exécution de programmes interagissant directement ou indirectement avec les utilisateurs finaux munis de leur mobile. Les tests de ce service doivent permettre de vérifier qu'une application est bien écrite et répond aux attentes des utilisateurs finaux. Dans cette section, nous avons donc proposé une méthode pour valider la couche applicative WAE (couche services et formats) et en particulier une application écrite en WML.

Le WAE est divisé en deux couches logiques :
- *la couche agents utilisateurs* : elle comprend des éléments comme les navigateurs, les répertoires téléphoniques, les éditeurs de messages,

[9]Wireless Application Protocol
[10]Markup Wireless Language

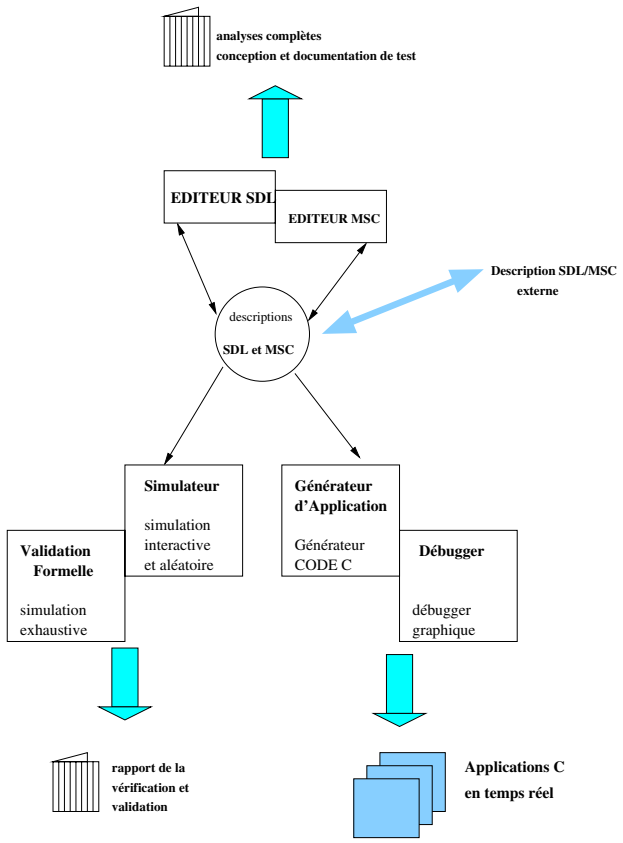


FIG. 4.2 – Environnement ObjectGEODE

etc.

– *la couche services et formats* : elle comprend des éléments courants et des formats accessibles aux agents utilisateurs, tels que les langages WML et WMLScript, les formats d'images, les formats vCard et vCalendar, etc.

4.3.1 L'architecture WAE

Le modèle logique WAE

L'architecture WAE comporte tous les éléments de l'architecture WAP relatif à la spécification et à l'exécution d'application. Cette architecture est surtout axée sur les aspects client de l'architecture du système WAP, notamment en ce qui concerne les agents utilisateurs. L'architecture WAE est définie pour l'essentiel en termes de concepts de réseau, formats de contenu, langage de programmation et services partagés. Les principaux éléments du modèle WAE comportent :

– *les agents utilisateurs* : ce sont des logiciels clients, inclus dans le terminal, qui offrent à l'utilisateur final des fonctionalités spécifiques (par ex., l'affichage de contenu).
– *les générateurs de contenu* : il s'agit d'applications ou de services intégrés sur les serveurs d'origine (par ex., les scripts CGI) qui produisent des formats de contenu standard en réponse aux requêtes d'agents utilisateurs du terminal mobile.
– *l'encodage standard de contenu* : l'encodage standard de contenu comprend l'encodage compressé pour le WML, l'encodage intermédiaire pour le WMLScript, les formats standards d'images, etc..
– *le WTA (Wireless Telephony Application)* : ensemble d'interfaces permettant d'accéder à différentes fonctions de téléphonie d'un terminal comme par exemple la composition d'un numéro.

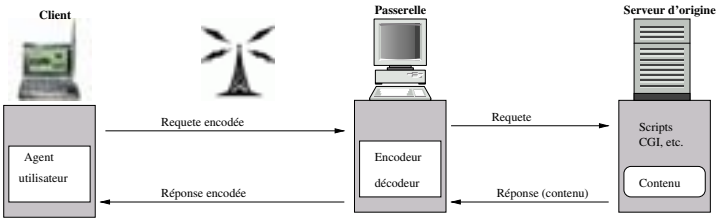


FIG. 4.3 – Modèle logique WAE

Le WML

Le WML est dérivé du XML[11]. Comme le HTML[12], il se compose de balises (ou tags), entre crochets, qui spécifient différents attributs et leurs valeurs (saisies entre guillemets). Les balises fonctionnent toujours par paire (un marqueur d'ouverture et un marqueur de cloture) et encadrent ainsi le texte qu'elles modifient. Le WML utilise la métaphore d'une carte (card) et d'un jeu de cartes (deck). Il contient des structures qui permettent à l'application de spécifier des documents constitués de plusieurs cartes. Une interaction avec un utilisateur est décrite dans un ensemble de cartes, qui peuvent être regroupées dans un document communément désigné sous le nom de "deck". Logiquement, un utilisateur navigue à travers un *deck* WML, explore une carte, visualise son contenu, peut entrer des informations à la demande et effectuer des choix, puis il se déplace vers une autre carte.

WML comporte un grand nombre de fonctionnalités, parmi lesquelles :
– *support texte et images* : le WML fournit aux auteurs le moyen de

[11]Extensible Markup Language
[12]Hypertext Markup Language

définir le texte et les images à présenter à l'utilisateur ; peuvent s'y ajouter des directives de disposition et de présentation ;
- *support des saisies de l'utilisateur* : le WML propose plusieurs éléments permettant de susciter une saisie de données par l'utilisateur ;
- *pile de navigation et d'historique* : le langage WML autorise plusieurs mécanismes de navigation se basant sur les URL et présente également un mécanisme d'historique de premier ordre. La navigation exploite les liens hypertexte, des éléments de navigation inter-cartes, ainsi que des éléments d'accès à l'historique de navigation.

4.3.2 Méthodologie de test

L'objectif est la génération automatique des scénarios de test pour un programme écrit en WML (deck WML). Notre méthodologie repose en partie sur l'obtention d'une spécification SDL représentative du comportement de l'application écrite en WML. Celle-ci se déroule en cinq étapes qui sont :
- proposition d'une architecture de test ;
- génération de l'arbre de comportement de l'application WML ;
- sauvegarde de l'arbre de comportement en automate ;
- traduction de l'automate en modèle SDL ;
- choix des objectifs de test ;
- génération de test avec les méthodes classiques.

Il convient de signaler ici que la traduction de l'automate en modèle SDL se fait encore à la main, l'objectif étant d'automatiser cette tâche. Ce qui nous permettra à terme de passer de l'application en WML au modèle SDL.

Architecture

La figure 4.4 (partie gauche) montre les différentes interactions entre l'agent utilisateur (sur le mobile) et l'application (le deck WML dans le cache du mobile). En observant les différents terminaux mobiles du marché, nous avons relevé les différentes possibilités d'interactions suivantes :
- visualisation du contenu des cartes (*écran*) ;
- déplacement à travers le menu et choix d'un sous-menu, d'une option, etc.. (*bouton central*) ;
- navigation à travers les cartes du deck et validation des sous-menu, options, choix, etc..(bouton *Accept*) ;
- retour à la carte précédente (bouton *Prev*) ;
- saisie des données (*clavier*).

La figure 4.4 (partie droite) présente l'architecture de test avec un PCO entre l'interface et le deck WML. Ce PCO va permettre d'observer et de contrôler les résultats de l'interaction entre l'interface et le programme.

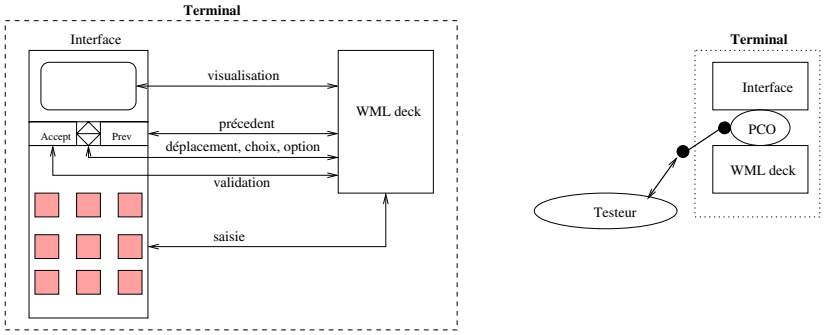


FIG. 4.4 – Modèle d'interaction entre l'interface et le deck WML et Architecture de test

4.3.3 Génération de l'arbre de comportement d'une application WML

Généralement, on part du modèle vers le code exécutable. Mais ici, notre objectif est de partir du code source (en langage WML) pour obtenir son équivalent SDL (reverse engineering) afin d'utiliser les outils classiques de génération de test qui prennent en entrée du SDL.
La première étape de notre méthode est la génération d'un automate qui représente le comportement de l'application WML. Pour ce faire, nous avons développé dans le cadre de ce travail, une application appelée *GenTree* (figure 4.5). *GenTree* est un outils qui permet de générer et de visualiser l'arbre de comportement d'une application écrite en WML. Son utilisation est facilitée par la présence d'une interface graphique conviviale comprenant une barre de menus et une barre d'outils qui permettent à l'utilisateur d'accéder aux différentes fonctionnalités du logiciel. Cette application prend en entrée une application WML et effectue :

- une analyse lexicale et syntaxique,
- développement et visualisation de l'arbre de comportement de l'application,
- sauvegarde de l'arbre de comportement sous forme d'automate d'états finis étendu (EFSM[13]).

Ces points sont détaillés dans les sous-sections suivantes.

Analyse lexicale et syntaxique

Cette analyse va permettre de voir si l'application est bien écrite en utilisant des spécifications correctes du WML.

[13]Extended Finite State Machine

Développement et visualisation de l'arbre de comportement

Le menu de la figure 4.5 nous permet d'ouvrir aisément le fichier dont on veut observer l'arbre de comportement. Après ouverture du fichier, les premiers noeuds de l'arbre vont apparaître, l'utilisateur peut alors décider de visualiser ou non les fils de chaque noeud (qui n'est pas une feuille) en appuyant sur la petite clé se trouvant à l'extrémité gauche de chaque noeud.

La sélection d'un noeud correspond à un élément de lien (tout élément

FIG. 4.5 – Inteface graphique de GenTree

comprenant un attribut qui permet d'effectuer un lien vers une autre carte tels *ontimer, onpick, href*, etc..) permet d'ajouter la carte à laquelle se fait ce lien à la liste des noeuds fils du noeud sélectionné, cette sélection s'effectue en cliquant une seule fois sur le noeud en question.

L'utilisateur peut ainsi développer à sa guise cet arbre en cliquant sur la clef à l'extrême gauche de chaque noeud ou en sélectionnant un noeud de l'arbre représentant un élément de lien.

Il se peut que le lien sélectionné branche vers une carte déjà visitée, dans ce cas, cette carte est automatiquement entourée par un rectangle bleue qui distingue cette dernière des autres cartes. Par la suite, une boîte de dialogue apparaît pour demander à l'utilisateur s'il désire retourner à cette carte, continuer de développer l'arbre ou ne rien faire (en appuyant sur le bouton *Cancel*).

(a) Sauvegarde de l'arbre (b) Option des attributs

FIG. 4.6 – Sauvegarde et Option des attributs

Le menu *Option* (second menu dans la barre des menus), comprend plusieurs options qui permettent de paramétrer la visualisation de l'arbre de comportement d'une application écrite en WML. Nous abordons dans ce qui suit chacune des options :

- *icone* : ce menu permet à l'utilisateur de choisir les types d'îcones qu'il souhaite utiliser pour la visualisation de l'arbre, deux possibilités sont offertes.
- attributs : chaque noeud de l'arbre autre que les noeuds texte représente un élément *WML* suivi d'une suite d'attributs (qui peut éventuellement être vide). Ainsi, comme le montre la figure 4.5, le premier élément *card* est suivi des attributs *id, ontimer, etc..*; à chacun de ces attributs est associée une valeur unique entre guillemets.
  Les éléments de *WML* pouvant avoir une longue liste d'attributs (pas tous très important), cela peut nuire à la clarté de la visualisation. d'où l'idée de laisser l'utilisateur le choix entre trois possibilités (figure 4.6) : afficher tous les attributs, afficher certains attributs ou n'afficher aucun attribut.
- *noeud* : l'utilisateur a aussi la liberté de visualiser ou de masquer certains types de noeuds (p, text, anchor, etc..) en sélectionnant ou non les cases à cocher associées.
- *arbre* : Le sous-menu *Tree* du menu *Option* laisse aux utilisateurs le choix entre deux manières de visualiser l'arbre de comportement (figure 4.7) :
  - Visualiser partiellement l'arbre : c'est l'option par défaut. Ici, l'utilisateur développe progressivement l'arbre en sélectionnant les noeuds contenant un lien. Cette option permet de simuler le déroulement de l'application WML.
  - Visualisation de l'arbe complet : en sélectionnant cette option, l'arbre global de l'application s'affiche automatiquement. Cette option est très utile surtout si l'utilisateur désire sauvegarder l'arbre global de son application dans un fichier texte.

(a) Noeuds à visualiser (b) Arbre

FIG. 4.7 – Option des noeuds et présentation de l'arbre

Sauvegarde de l'arbre de comportement sous forme d'automate

Après avoir générer l'arbre de comportement relatif à l'application WML, *GenTree* nous adonne la possibilité de sauvegarder (figure 4.6)cet arbre de comportement afin d'en faire un usage ultérieur. Cette sauvegarde peut se faire sous trois fromes :

- sauvegarde sous forme d'automate : *GenTree* permet aussi la sauvegarde de l'automate associé à l'arbre de comportement d'une application sous forme d'une suite de triplets (card1, [evt], card2) où *evt* représente l'évènement qui assure le passage de la carte *card1* à la carte *card2*.
- sauvegarde sous forme de liste d'adjacence : l'arbre de comportement peut être sauvegardé sous forme d'un graphe représenté par un liste d'adjacence. Dans cette représentation, l'arbre est parcouru en profondeur (à gauche d'abord) et pour chaque noeud visité on sauvegarde son identificateur suivi de la liste des identificateurs de tous les noeuds fils séparés par ";".
- sauvegarde sous forme de phrase parenthésée : grâce à un parcours en profondeur de l'arbre, ce dernier peut être représenté d'une façon unique sous forme d'une expression parenthésée comprenant des crochets('[' et ']'), parenthèses et des virgules pour séparer les différents éléments. Cette représentation a pour avantage d'économiser de l'espace mémoire mais s'avère peu lisible.

4.3.4 Translation de l'automate en modèle SDL

L'application développé (*GenTree*) nous a permis de développer un arbre de comportement d'une application WML et de la sauvegarde de celui-ci. Le format de sauvegarde (automate, liste d'adjacence, phrase parenthésée) ne nous convient pas en vue de la génération automatique de tests. Nous allons donc traduire le comportement obtenu en modèle SDL et la sauvegarde sous forme d'automate va nous faciliter la tâche.

A cet effet, il existe des travaux [10] qui montrent comment transformer un EFSM en SDL. La figure 4.8 montre la traduction d'un EFSM de base en SDL. Une transition est représentée par un signal d'entrée, une tâche et une séquence de signaux de sortie en SDL. Dans ladite figure, lorsque l'état $s1$ reçoit un évènement d'entrée avec un paramètre p, il réalise la tâche $action2$ et la sortie $output2$. Sur la base de cette technique de traduction, nous déduisons de proche en proche le modèle SDL conformément à l'automate qui décrit l'application en WML.

Après obtention de la spécification SDL, les procédés classiques nous per-

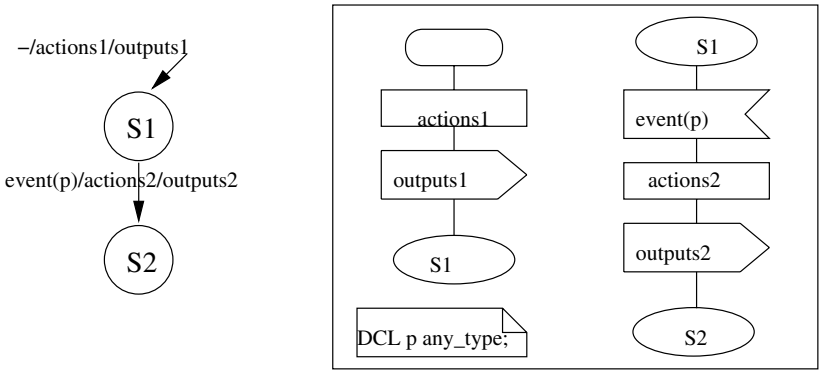


FIG. 4.8 – Translation d'un EFSM en SDL

mettent de générer les séquences de test. Comme objectif, il peut être intéressant de voir si une carte particulière peut être atteinte. La simulation peut aussi révéler des situations d'interblocage (deadlock) et/ou d'attente infinie (livelock). Dans le cas d'une attente infinie, ça veut dire que sur une carte donnée, il est impossible de continuer la navigation vers d'autres cartes.

L'intérêt de notre méthodologie est de pouvoir générer un automate pour n'importe quelle application WML, contrairement à la méthode appliquée dans [59] qui effectue une génération de test pour une application donnée. Cette technique est aussi intéressante dans le cas du test de régression, en cela, elle va permettre de vérifier qu'une application WML à laquelle est ajoutée des fonctionnalités satisfait encore les spécifications d'avant modification.

4.4 Spécification et génération de test pour les couches protocolaires WSP et WTP

Les spécifications du WAP définissent un ensemble de protocoles pour les couches transport, session et application afin de permettre aux opérateurs

et aux constructeurs de répondre rapidement et avec souplesse aux besoins croissants en matière de services différenciés et avancés.
Pour la modélisation de la pile protocolaire WAP, nous nous sommes focalisés sur les couches WSP et WTP et avons étudié le système pour la transmission des messages en mode connexion.

4.4.1 Les classes de transaction

Cette sous-section décrit les classes de transaction WTP. Les classes de transaction sont définies par l'initiateur, et indiquées dans le message de requête envoyé au destinataire.

- classe 0 : c'est la classe des requêtes non fiabilisées sans réponse. L'initiateur envoie une requête au destinataire. Le destinataire n'informe pas de la réception de la requête ; l'initiateur n'en effectue aucune retransmission. Côté initiateur, la transaction prend fin après l'envoi de la requête. Côté destinataire, la transaction prend fin après réception de la requête ;
- classe 1 : c'est la classe des requêtes fiabilisées sans réponse. L'initiateur émet une requête au destinataire qui en accuse réception. Côté initiateur, la transaction prend fin une fois l'accusé de réception reçu. La transaction peut être interrompue à tout moment.
- classe 2 : c'est la classe des requêtes fiabilisées avec réponses fiabilisées. Les transactions de cette classe constituent le mécanisme de base de la transaction requête/réponse. Une session WTP pourra comporter plusieurs transactions de ce type. L'initiateur envoie une requête au destinataire. Le destinataire envoie une réponse unique qui accuse implicitement réception du message de requête. Le destinataire renvoie ensuite la réponse à l'initiateur qui en accuse réception. Côté destinataire, la transaction prend fin une fois l'accusé de réception reçu. La transaction peut être interrompue à tout moment.

4.4.2 Modélisation de la couche protocolaire WTP

Le *WTP* est un protocole de transport qui s'appuie sur un service de datagrammes, éventuellement sécurisé. Il a été pensé comme un protocole simple de transactions, qui convient pour des clients disposant de peu de ressources (stations mobiles) et qui est d'une bonne efficacité sur les réseaux sans fil. Les avantages liés à l'utilisation du WTP sont les suivants :

- amélioration de la fiabilité en matière de transport de datagramme : le *WTP* exonère la couche supérieure des réexpéditions et accusés de réception requis pour assurer la fiabilité de transmission des datagrammes,
- accroissement de l'efficacité sur les services orientés connexion : le *WTP* ne comporte aucune configuration de connexion explicite, ni

phase de fin de communication,
- enfin, le *WTP* est orienté message et est conçu pour des services en mode requêtes/réponses, tels que la "navigation".

Spécification du système

Un protocole WTP est constitué des modules (des processus dans le modèle SDL) *initiateur* et *destinataire*. Les modules *initiateur* traitent des transactions qui sont initiés dans le protocole, les modules *destinataire* traitent des transactions initiées dans un protocole distant. Il y a un processus par transaction dans chaque protocole. Les communications avec les autres protocoles se font :

– avec la couche WSP cliente du protocole WTP,
– avec le protocole WDP. Comme seules les couches WSP et WTP sont formalisées dans notre modèle, les signaux émis et reçus par les protocoles WTP vers et provenant de WDP sont directement échangés avec l'environnement.

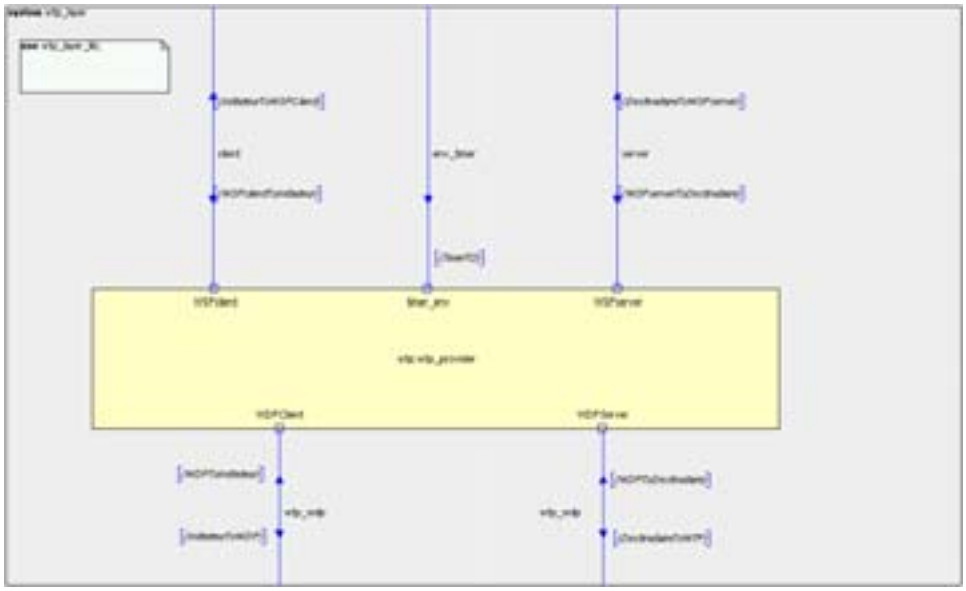


FIG. 4.9 – Système WTP

La structure d'une architecture constituée uniquement de la couche WTP (avec uniquement un protocole WTP) correspond à la figure 4.9. Le système est constitué du bloc *wtp_provider*. Ce bloc interagit avec l'environnement (couches WSP et WDP) via les sigaux suivants de type *signallist*[14] :

[14]Un *signallist* spécifie un ensemble de signaux déclarés préalablement

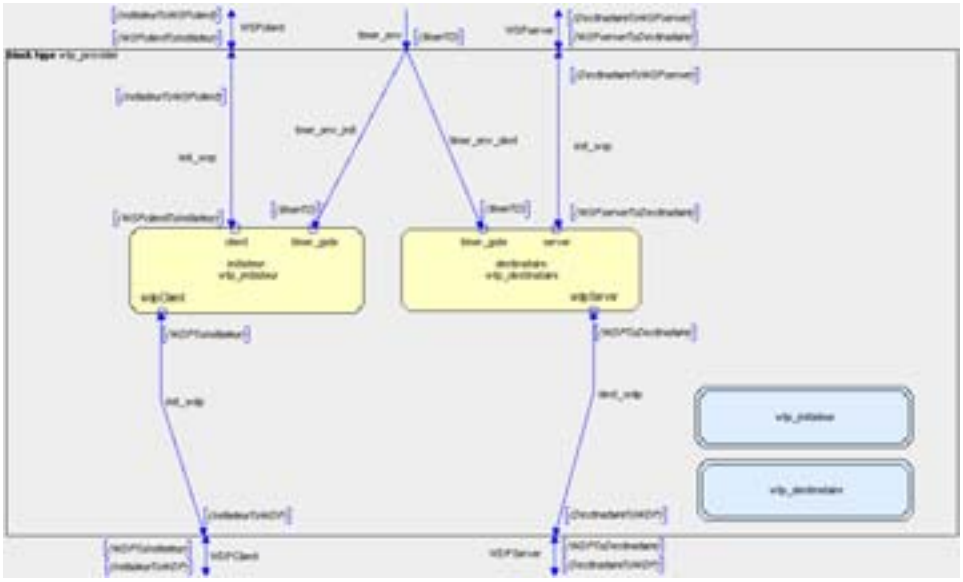


FIG. 4.10 – Bloc WTP Provider

- *initiateurToWSPclient* : c'est l'ensemble des signaux qui partent du processus initiateur WTP vers le processus client WSP,
- *WSPclientToInitiateur* : c'est l'ensemble des signaux qui partent du processus client WSP vers le processus initiateur WTP,
- *DestinataireToWSPserver* : c'est l'ensemble des signaux qui partent du processus destinataire WTP vers le processus serveur WSP,
- *WSPserverToDestinataire* : c'est l'ensemble des signaux qui partent du processus serveur WSP vers le processus destinataire WTP,
- *initiateurToWDP* : c'est l'ensemble des signaux qui partent du processus initiateur WTP vers le processus WDP,
- *destinataireToWDP* : c'est l'ensemble des signaux qui partent du processus destinataire WTP vers le processus WDP,
- *WDPToInitiateur* : c'est l'ensemble des signaux qui partent du processus WDP vers le processus initaiteur WTP,
- *WDPToDestinataire* : c'est l'ensemble des signaux qui partent du processus WDP vers le processus destinataire WTP.

Le bloc wtp_provider est composé des états et processus suivants :

- processus wtp_initiateur avec les états *null, result_wait, result_resp_wait et wait_timeout*,
- processus wtp_destinataire avec les états *null, result_wait, wait_timeout et result_resp_wait.*

4.4.3 Temporisation, compteurs et variables

Nous présentons dans cette sous-section les temporisateurs, les compteurs et les variables qui définissent entre autre le noyau de l'automate du protocole *WTP*.

Temporisations

Chaque transaction comporte une temporisation associée. Cette temporisation est utilisée à la fois pour l'intervalle de réémission, l'intervalle d'accusé de réception et l'intervalle de dépassement de temps d'attente. Les valeurs de temporisation sont regroupées en fonction du but qui leur est assigné, c'est ainsi qu'on a :

- A : temporisation d'accusé de réception, elle définit une limite de durée d'attente avant l'envoi de l'accusé de réception,
- R : temporisation de réémission, elle définit une limite de durée d'attente avant la réexpédition d'un message,
- W : temporisation de dépassement de temps d'attente, elle définit une limite de durée d'attente avant la diffusion de l'état d'une transaction.

Compteurs

Les compteurs suivants sont utilisés par le WTP :

- RCR : compteur de réexpédition, il définit une limite pour le nombre maximal de réexpéditions d'un message quelconque. La valeur maximale est définie par CR_MAX,
- AEC : compteur d'expiration de l'accusé de réception, il définit une limite pour le nombre maximal de fois où la temporisation de transaction, initialisée avec l'intervalle d'accusé de réception, est autorisée à se relancer avant que la transaction ne doive être abandonnée. La valeur maximale est définie par AEC_MAX.

Variables

Les variables suivantes sont utilisées par l'utilisateur WTP côté initiateur et côté destinataire.

- GenTID : de type Uint16[15], identificateur à utiliser pour la prochaine transaction. Incrémenté à chaque nouvelle transaction,
- SendTID : de type Uint16, valeur d'identificateur à inscrire dans tous les messages de cette transaction,
- RcvTID : de type Uint16, valeur d'identificateur à inscrire dans tous les messages de cette transaction,
- LastTID : de type Uint16, dernier identificateur reçu d'un hôte distant quelconque,

[15]Le type Uint16 est un entier non signé sur 16 bits.

- HolOn : de type BOOL[16], VRAI si l'accusé de réception de mise en attente a été reçu,
- Uack : de type BOOL, VRAI si l'accusé de réception utilisateur a été demandé pour cette transaction.

4.4.4 Vérification et validation du modèle

La conception du modèle constitue la moitié du travail de vérification et validation [21] qui a pour objectif de détecter les erreurs de conception (deadlock, etc..) et la vérification du comportement du modèle par rapport à certaines propriétés. La démonstration d'un comportement correct doit être réalisée à travers des sessions de simulation.
Nous avons utilisé le simulateur ObjectGEODE (OG) dans les trois modes de simulation qui sont : aléatoire, interactive et exhaustive. Les erreurs de conception peuvent être détectées et les propriétés du modèle vérifiées dynamiquement dans les modes interactifs et aléatoires. Mais pour avoir une entière confiance dans le modèle, une vérification dynamique qui couvre tous les comportements du modèle s'avère nécessaire, en cela le mode exhaustif constitue un atout malgré le risque d'explosion combinatoire, nous montrons par la suite comment limiter ce risque.
Le mode interactif est aussi intéressant pour montrer comment le modèle se comporte lorsqu'on exécute un scénario spécifique. Mais la vérification des propriétés est essentielle pour les systèmes et est le domaine de la simulation exhaustive

Paramètres du simulateur

- *environnement du système* : avec le simulateur d'OG, un ensemble de commandes dite "feed" permet à la session de simulation de consommer les transitions dont les signaux d'entrée sont produits par l'environnement. Dans notre modèle, la liste de ces signaux est décrite par les lignes ci-dessous (les parties droite et gauche représentent respectivement les canaux de communication et les signaux qu'ils supportent) :

```
env_timer timerto
env_timer timerto_a
env_timer timerto_r
env_timer timerto_w
env_timer timerto_r
env_timer timerto_a
wtp_wdp rcvack
wtp_wdp rcvabort
```

[16]Le type BOOL est une valeur booléenne qui ne peut contenir que la valeur VRAI ou FAUX.

```
wtp_wdp rcvresult
wtp_wdp rcvinvoke
wtp_wdp rcvack
wtp_wdp rcvabort
client tr_result.res
client tr_invoke.req
client tr_abort.req
server tr_result.req
server tr_invoke.res
server tr_abort.req
```

– *réduction de la taille d'espace des états* : le principal inconvénient d'une simulation exhaustive est la potentielle explosion de l'espace des états. Pour réduire la taille du graphe complet, différentes techniques peuvent être utilisées - une des plus efficaces consiste à ré-initialiser les variables du processus à la fin de chaque transition. Dans notre modèle, nous avons utilisé des *MSC observers*. Il s'agit des scénarios ou expression de propriétés compilés en même temps que le modèle SDL. Ils permettent de vérifier qu'une séquence d'évènement est bien fournie par le modèle SDL et la détection de violation de la séquence exprimée. Vu qu'il se traduit comme condition d'arrêt de la simulation, il a donc comme intérêt de raccourcir la simulation dès que les conditions/propriétés exprimées sont satisfaites ou violées.

Simulation interactive

Pour exercer le modèle, différentes simulations simples ont été exécutées :
– *scénario 1* : la figure 4.11 présente le scénario d'une transaction complète et réussie de classe 2 (du point de vue du processus destinataire),
– *scénario 2* : la figure 4.12 présente le scénario d'une transaction avec expiration du timer *aec*. *aec* est le compteur d'expiration d'accusé de réception. En effet le timer expire et s'en suit une nouvelle réception d'invocation de méthode (*rcvinvoke*) ;
– *scénario 3* : la figure 4.13 présente le scénario d'une transaction de classe 2 avec interruption de celle-ci par le destinataire. Le destinataire renvoie une réponse qui accuse réception du message (*ack_pdu*) de requête et en lieu et place de la réponse, il renvoie un message d'interruption (*abort*).

Vérification de propriétés

– *propriété* : nous avons vérifié la propriété de transaction de classe 2 qui stipule que la transaction peut être interrompue à tout moment. Celle-ci est représentée par un *MSC* à la figure 4.14 ;

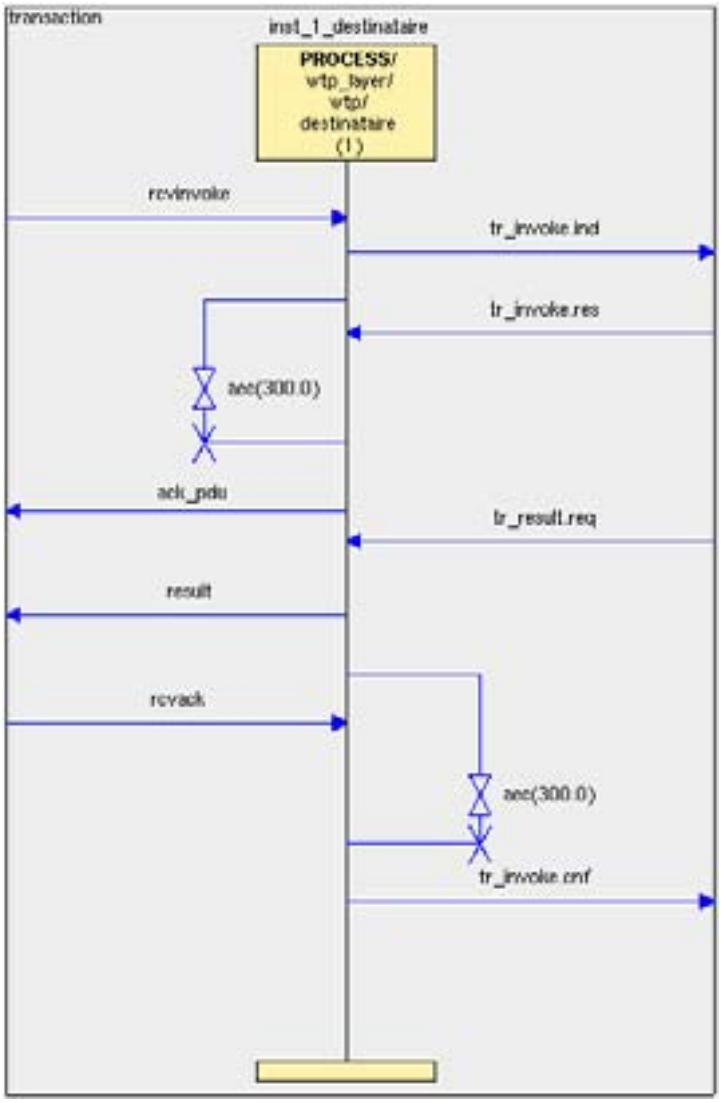


FIG. 4.11 – Scénario d'une transaction complète

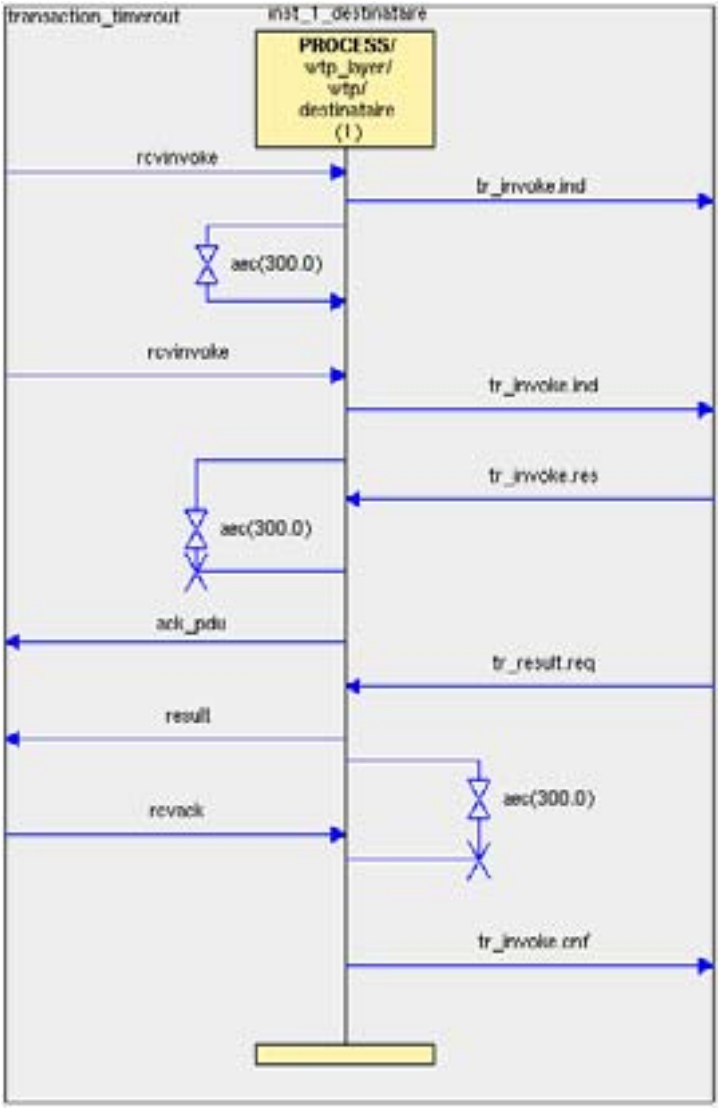


FIG. 4.12 – Scénario d'une transaction avec expiration du timer

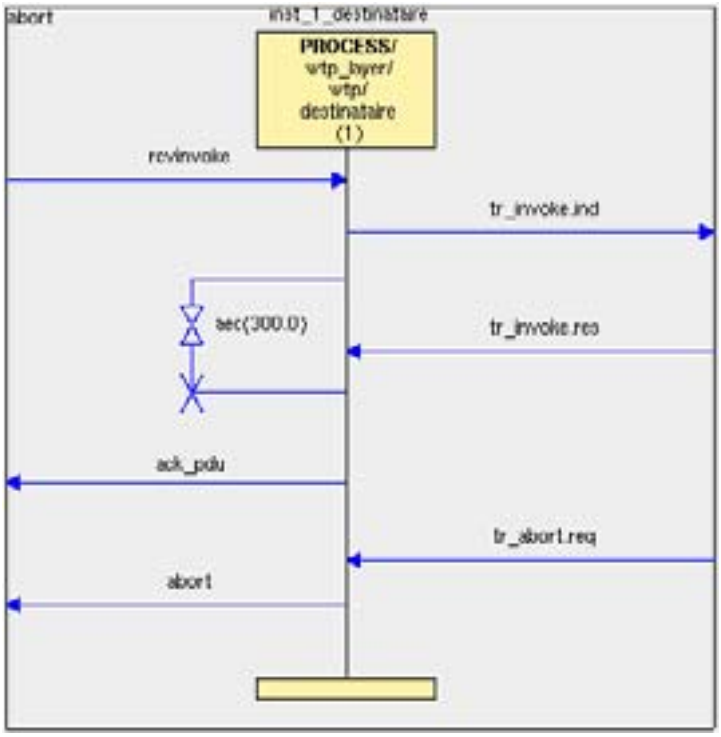


FIG. 4.13 – Scénario d'une transaction avec interruption du destinataire

– Option de la simulation exhaustive :
  Par défaut, le mode d'exploration du graphe d'états se fait en largeur (breadth-first). Cela signifie que toutes les transitions sont consommées pour un état donné avant l'exploration d'un nouvel état. Cependant, pour un graphe d'état de taille importante, l'usage de l'exploration en profondeur est plus approprié pour détecter les erreurs de conception. Les options relatives au nombre d'états et à la profondeur atteinte (state limit, depth limit) contrôlent la longueur de la simulation. Nous limitons la profondeur de notre simulation à 35. Cela signifie que la simulation arrêtera l'exploration des séquences dans lesquelles 35 transitions ont été exécutées. Cette longueur de 35 doit être comparée avec la longueur nominale du scénario relatifs à la propriété 1. Dans notre cas il est égal à 17, incluant les 5 étapes initiales. Notre limite correspond alors au double de la longueur nominale.

Compte tenu des options décrites précédemment et de la propriété à vérifier, la simulation exhaustive de notre modèle a donné les résultats suivants :

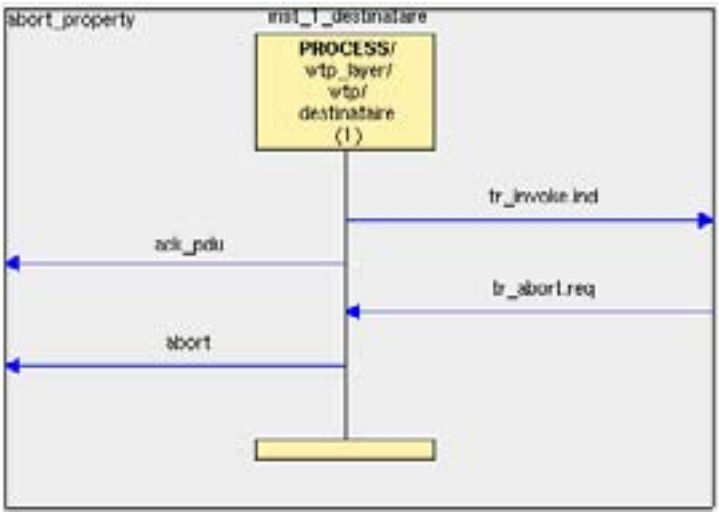


FIG. 4.14 – Propriété relative à l'abandon d'une transaction

- *graphe d'état* : environs 17582 états (nombre de différents états atteints), 69164 transitions (nombre de transitions consommées, soit une moyenne d'environs 3,9 transitions par état), avec une profondeur (profondeur maximale atteinte lors de la recherche) de 15 transitions consécutives et une largeur maximale de 6888 états ;
- 100% de transitions et 100% d'états ont été respectivement consommées et atteints ;
- 2260 succès ont été détectés, en d'autres termes, aucune violation de la propriété n'a été décelée ;
- aucune exception, aucun deadlock n'ont été décelés, généralement (mais pas nécessairement) ils correspondent à des erreurs de conception ;
- la simulation s'est arrêtée après que la profondeur maximale ait été atteinte.

Nous retenons que 100% de transitions et d'états ont été consommées et atteints pour la propriété vérifiée. La simulation révèle une absence de blocage et la cohérence entre la spécification SDL et le comportement souhaité du système. Cependant, nous avons relevé un certains nombre d'ambiguïtés qui pourraient être à l'origine de problèmes d'interopérabilité. Nous avons donc sur ces points interprété la norme, en particulier dans les comportements suivants :

– le standard ne spécifie pas le code d'erreur du *TR-abort* envoyé vers la couche supérieure en cas de débordement des temporisateurs,
– le standard ne spécifie pas (ou laisse à l'implantation) le comportement du protocole lors de la réception d'un signal sur un état pour lequel ce signal n'est pas attendu. Ce genre de comportement peut ce-

```
(8192 states 28531 transitions 3 seconds, depth=13, breadth=2756)
(16384 states 62780 transitions 6 seconds, depth=14, breadth=4477)
verify stopped by depth limit

Number of states : 17582
Number of transitions : 69164
Maximum depth reached : 15
Maximum breadth reached : 6888
duration : 0 mn 6 s

Number of exceptions : 0
Number of deadlocks : 0
Number of stop conditions : 0
Transitions coverage rate : 100.00 (0 transitions not covered)
States coverage rate : 100.00 (0 states not covered)
Basic blocks coverage rate : 100.00 (0 basic blocks not covered)
Number of errors : 0
Number of success : 2260
```

FIG. 4.15 – Propriété 2 : Résultat de la simulation

pendant se produire assez souvent, en partie dû à l'aspect asynchrone des échanges.

– le standard ne spécifie pas non plus (ou laisse à l'implantation) les états d'arrivée lors de la réception de certains signaux. C'est le cas du processus initiateur (pour ne citer que celui-ci dans les états suivants :
  - état null, sur réception de TR_Invoke.ind avec le paramètre Uack = true,
  - état result_wait sur réception de RcvResult avec le paramètre HoldOn = false,
  - état result_resp_wait sur réception de TR_Abort.req,
  - état wait_timerTO_W sur réception de TR_Abort.req.

4.4.5 Modélisation de la couche protocolaire WSP

Le *WSP* fournit aux applications de niveau supérieur une interface bien définie vers deux mécanisme de session. Le premier est un service en mode connecté (celui auquel nous nous intéressons) qui s'appuie sur le service de transaction *WTP*, le second est un service en mode non connecté s'appuyant sur une couche de transport sécurisé ou non en mode datagramme. Ce protocole offre les services les plus adaptés au support d'applications de navigation et est optimisé pour une utilisation sur des réseaux à débit réduit et à temps de latence élevé.

Spécification du système

La couche WSP est constituée de deux protocoles de comportements différents : *WSP-serveur* et *WSP-client*.

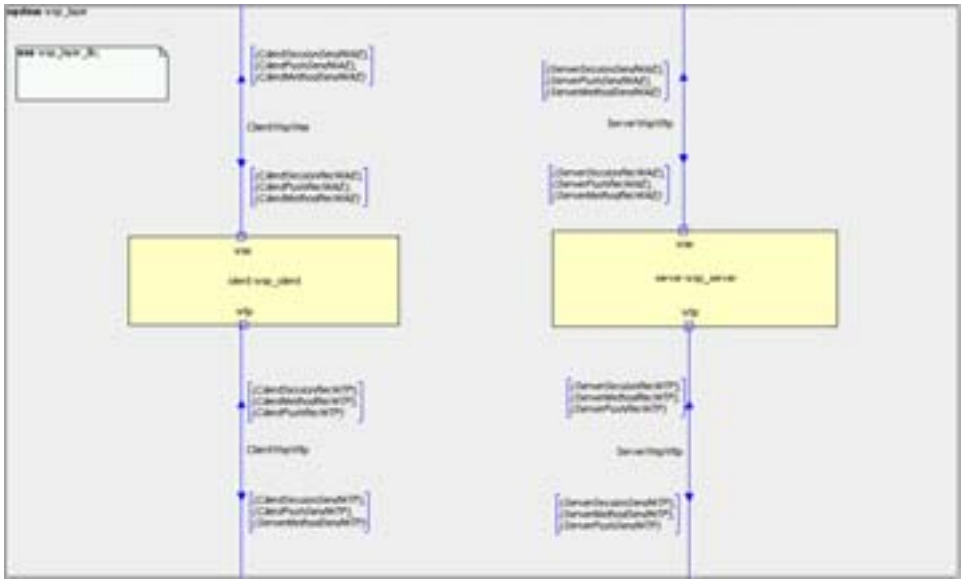


FIG. 4.16 – Système WSP

Le service de "session en mode connecté" est divisé en sous-services. La plupart de ces sous-services sont asymétriques et, de ce fait, les opérations possibles pour un client et un serveur connectés à la même session seront différentes. Les ensembles fonctionnels disponibles sont les suivants :

- *gestion de session* : Il assure la phase de connexion d'un client à un serveur en négociant les sous-services et les options de protocoles utilisables par les deux parties.
- *invocation de méthodes* : permet au client de demander au serveur d'effectuer des opérations pour lesquelles la connexion a été établie.
- *restauration de session* : inclut un moyen pour suspendre la session de sorte que son état soit préservé, les deux homologues de bout en bout s'accordant sur le fait qu'aucune nouvelle transmission ne peut avoir lieu tant que le client ne restaure pas la session.
- *push* : autorise un serveur à envoyer de l'information sans requête préalable d'un client en tirant parti du contexte de session partagé entre le client et le serveur. Ce sous-service est dit *"non confirmé"*, en ce sens qu'il n'y a pas de garantie que l'information poussée soit effectivement et correctement reçue.
- *push confirmé* : est similaire au précédent, mais il est demandé au client d'accuser réception de l'information envoyée. Le client peut aussi

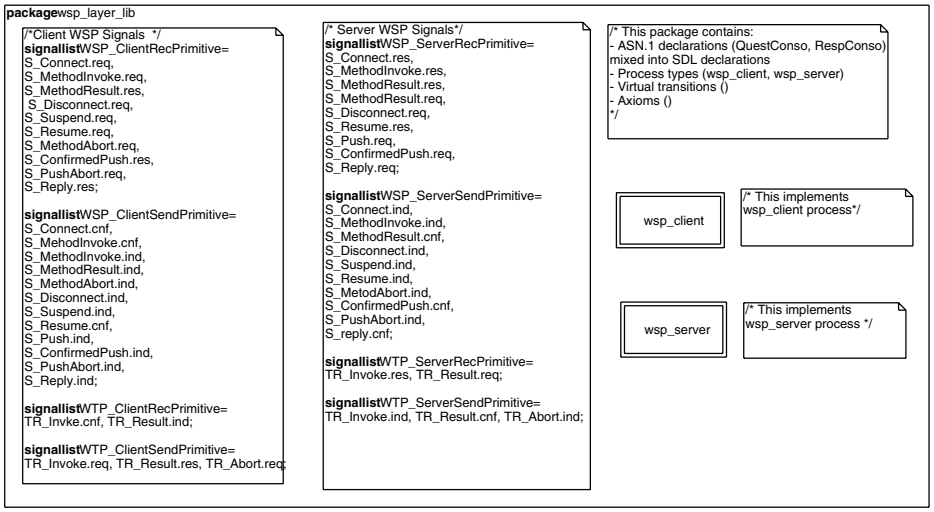


FIG. 4.17 – Package représentant une partie des signaux et bloc utilisés par le système

refuser le push, ce qui sera mentionné au serveur.

Conformément à la description du standard WSP, la structure interne (voir figure 4.18) est composée de processus gérant les sessions (*session*), de processus gérant les méthodes associées à une session (*method*), de processus gérant les push (*push*) et push confirmé (*confirmed-push*). De même, un service gérant une session reçoit les messages le concernant, parmi ces messages ceux concernant plus spécifiquement une méthode sont envoyés au service qui gère la méthode et ceux concernant plus spécifiquement un push (ou un push confirmé) sont envoyés au processus qui gère le push (ou push confirmé).

La structure d'une architecture constituée uniquement de la couche WSP (avec uniquement un protocole WSP) correspond à la figure 4.16. Le système est constitué des *blocs wsp_client* et *wsp_serveur*. Ces blocs interagissent avec l'environnement (couches WAE et WTP) via les sigaux suivants (voir figure 4.17) de type *signallist*[17] :

- *ClientSessionRecWAE, ServeurSessionRecWAE* : c'est l'ensemble des signaux qui partent de la couche WAE vers le processus session du client et du serveur WSP respectivement,
- *ClientSessionSendWAE, ServeurSessionSendWAE* : c'est l'ensemble des signaux qui partent des processus session du client et du serveur WSP vers la couche WAE ;

[17] Un *signallist* spécifie un ensemble de signaux déclarés préalablement

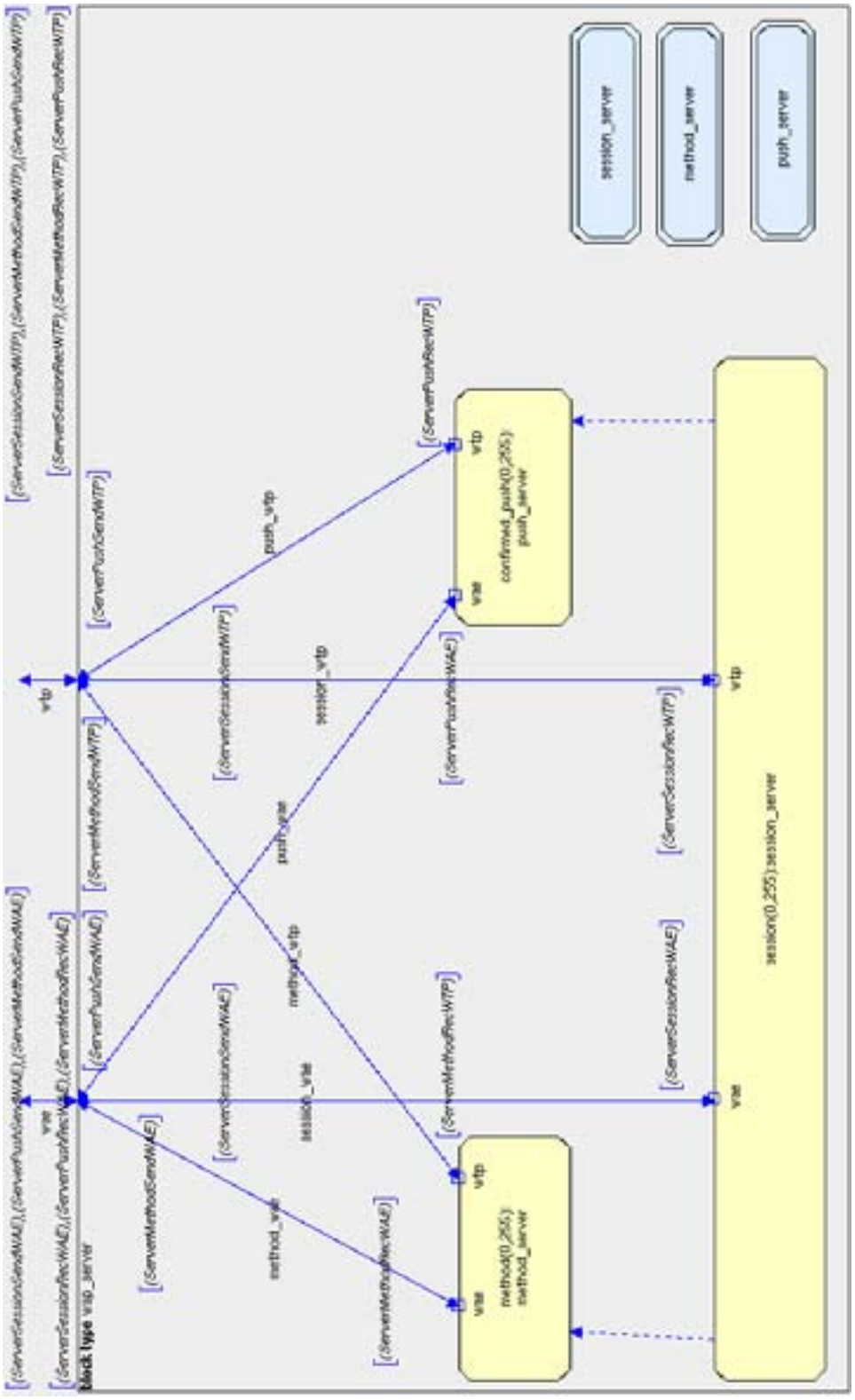


FIG. 4.18 – Bloc WSP côté serveur

- *ClientSessionRecWTP, ServeurSessionRecWTP* : c'est l'ensemble des signaux qui partent de la couche WTP vers le processus session du client et du serveur WSP respectivement,
- *ClientSessionSendWTP, ServeurSessionSendWTP* : c'est l'ensemble des signaux qui partent des processus session du client et du serveur WSP vers la couche WTP ;

Chacun des blocs (wsp_client et wsp_server) est composé des états et processus suivants :

- bloc wsp_client :
  - processus session avec les états *null, connecting, connected, suspended et resuming*
  - processus method avec les états *null, requesting, waiting et completing*
  - processus push avec les états *null et receiving*

  avec les états *null, result_wait, result_resp_wait et wait_timeout*,
- bloc wsp_server :
  - processus session avec les états *null, connecting, terminating, connecting2, connected, suspended, resuming et resuming2*
  - processus method avec les états *null, holding, requesting, processing et replying*
  - processus push avec les états *null et pushing*

4.4.6 Vérification et validation du modèle

Paramètres du simulateur

- *environnement du système* : le fichier *feeds*[18] est constitué des signaux suivants :

```
serverwtp tr_result.cnf
serverwtp tr_result.cnf
serverwtp tr_invoke.ind
serverwtp tr_invoke.ind
serverwtp tr_invoke.cnf
serverwtp tr_invoke.cnf
serverwtp tr_abort.ind
serverwtp tr_abort.ind
serverwtp tr_abort.ind
serverwtp suspend session
serverwtp release method
serverwtp disconnect session
serverwtp abort method
```

[18] liste des messages stimuli envoyés par le testeur vers le système sous test

```
serverwtp abort confirmed_push
serverwspwtp s_resume.res
serverwspwtp s_push.req
serverwspwtp s_methodresult.req
serverwspwtp s_methodresult.req
serverwspwtp s_methodinvoke.res
serverwspwtp s_methodinvoke.res
serverwspwtp s_methodabort.req
serverwspwtp s_disconnect.req
serverwspwtp s_connect.res
serverwspwtp s_confirmedpush.req
serverwspwtp s_confirmedpush.ind
```

– *réduction de la taille d'espace des états* : nous avons utilisé la même technique que pour la couche WTP pour réduire la taille d'espace des états.

Simulation interactive

Pour exercer le modèle, nous avons simulé le scénario d'une session avec invocation de méthode, la figure 4.19 présente ledit scénario.

Vérification de propriétés

– propriété : nous avons vérifié la propriété relative à une session aboutie (côté serveur). Celle-ci est représentée par un *MSC* à la figure 4.20 (partie gauche) ;
– Option de la simulation exhaustive : Le mode d'exploration se fait en largeur (breadth_first). Nous limitons la profondeur de notre simulation à 14 (double de la longueur nominale du scénario relatif à la session, dans notre cas il est égal à 7 incluant les 3 étapes initiales). Le fichier suivant décrit les options de la simulation :

```
mode breadth
define edges_dump ''
define states_dump ''
deadlock limit 10
exception limit 10
stop limit 10
define stop_cut true
error limit abort_property 10
success limit abort_property 10
error cut abort_property = true
success cut abort_property = false
```

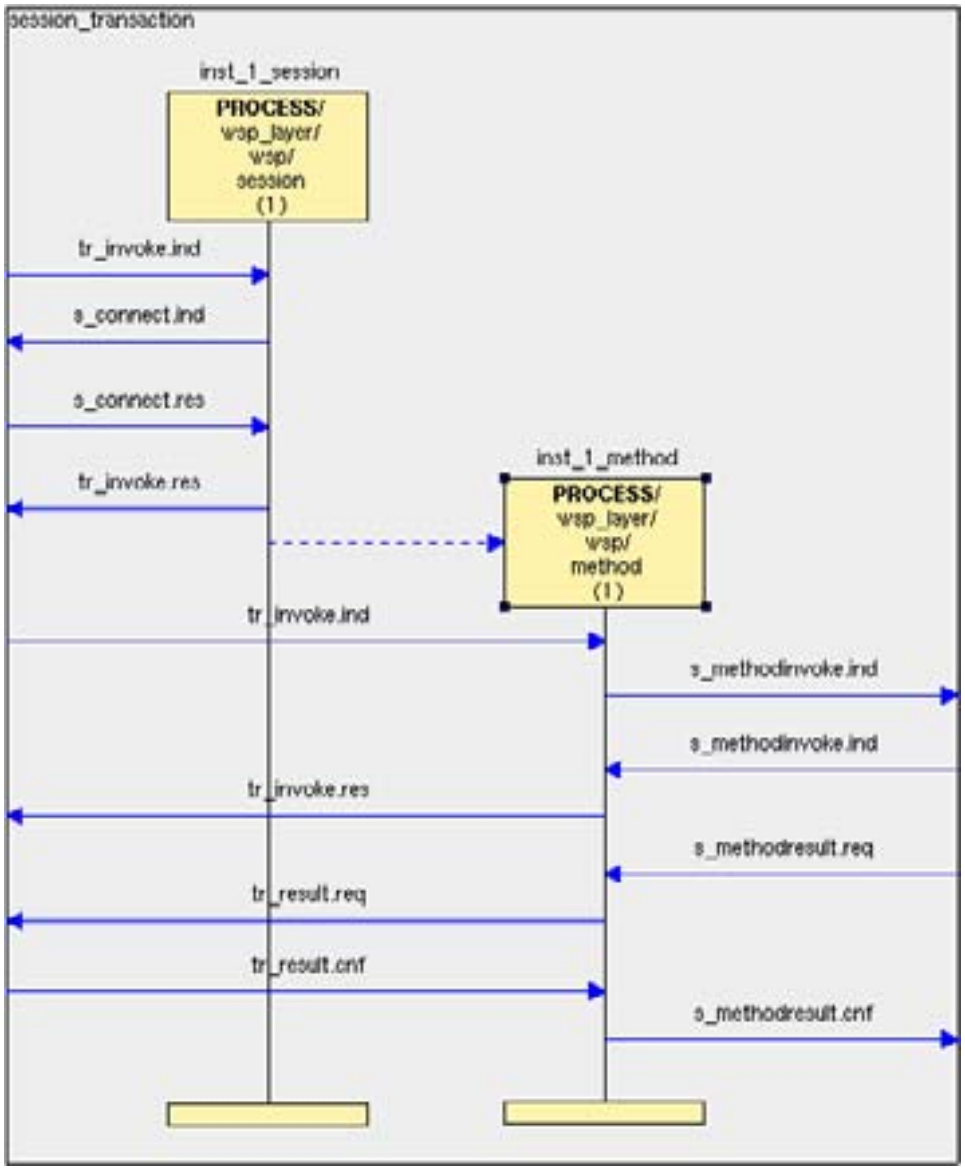
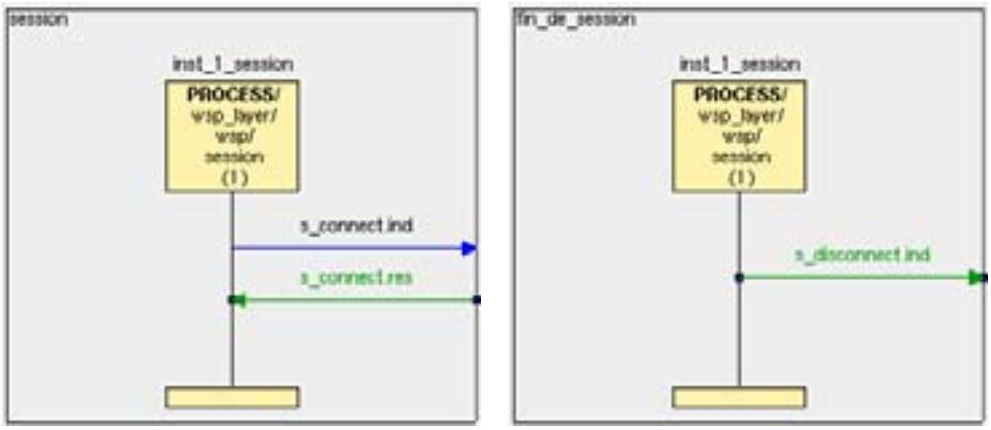


FIG. 4.19 – Scénario d'une session avec invocation de méthode



(a) Session (b) Fin de session

FIG. 4.20 – Connexion et déconnexion

```
define states_limit 0
define depth_limit 14
define depth_limit_stop false
define hash_fill_limit 10
define main_hash_size 100000
define hash_size 1000
define verify_stats false
```

Compte tenu des options décrites précédemment et des propriétés à vérifier, la simulation exhaustive de notre modèle a donné les principaux résultats suivants :

- *graphe d'état* : environs 26348 états (nombre de différents états atteints), 103742 transitions (nombre de transitions consommées, soit une moyenne d'environs 4 transitions par état), avec une profondeur (profondeur maximale atteinte lors de la recherche) de 14 transitions consécutives et une largeur maximale de 2325 états ;
- 100% de transitions et 100% des états ont été respectivement consommées et atteints ;
- 3422 succès ont été détectés, en d'autres termes, aucune violation de la propriété n'a été décelée ;
- aucune exception, ni deadlock n'ont été décelés, généralement (mais pas nécessairement) ils correspondent à des erreurs de conception ;
- la simulation s'est arrêtée après que la profondeur maximale ait été atteinte. Le graphe d'état ayant été en totalité exploré.

```
verify stopped by depth limit

Number of states : 26348
Number of transitions : 103742
Maximum depth reached : 14
Maximum breadth reached : 2325
duration : 0 mn 13 s

Number of exceptions : 0
Number of deadlocks : 0
Number of stop conditions : 0
Transitions coverage rate : 100 (0 transitions not covered)
States coverage rate : 100.00 (0 states not covered)
Basic blocks coverage rate : 100 (0 basic blocks not covered)
Number of errors : 0
Number of success : 3442
```

FIG. 4.21 – Résultat de la simulation : session aboutie

Comme pour la couche WTP, nous obtenons aussi des taux de couverture intéressant, une spécification cohérente en plus de l'absence de blocage. Nous avons aussi eu recours à l'interprétation de la norme dans certains cas à cause des ambiguïtés rencontrés. Par exemple, le standard ne spécifie pas (ou laisse à l'implantation) les états d'arrivée lors de la réception de certains signaux. C'est le cas du processus relatif à la session cliente dans les cas suivants :

- état *connecting*, sur réception des signaux *TR_Invoke.ind* et *TR_Invoke.cnf*,
- état *suspended*, sur réception du signal *TR_Invoke.ind*,
- état *resuming*, sur réception ddu signal *TR_Invoke.ind*,
- état *wait_timerTO_W*, sur réception de *TR_Abort.req* et *TR_Invoke.cnf*.

4.5 Spécification et génération de test pour les services de localisation

Comme nous l'avons annoncé dans l'introduction générale de ce document, un des challenges de l'Internet Mobile est de proposer des services innovant afin d'inciter les abonnés à l'adopter. A cet effet, les services basés sur la localisation ou géo-dépendant se positionnent comme une valeur de

différentiation pour les opérateurs. Ce service met en oeuvre des mécanismes qui permettent au serveur de localisation (LCS-serveur) de calculer les positions (information de localisation ou LCS-information) des mobiles afin de les fournir au serveur d'application (exemple un serveur WAP, LCS-client) sur demande autorisée. Le serveur d'application pourra ainsi renvoyer au terminal des informations personnalisées qui tiennent compte de sa position. Plusieurs entités hétérogènes du LCS-serveur interopèrent, tester ce service va donc permettre de garantir la fiabilité de la fourniture du LCS-information.

4.5.1 Spécification du système LCS services

La modélisation SDL des LCS services a été faite en tenant compte de l'architecture des services de localisation dans un réseau UMTS tels qu'ils sont décrits dans [57, 76, 74, 75] et le chapitre 2. Le système contient deux blocs fonctionnels qui sont :

- le bloc *réseau coeur* (core-network) : c'est la partie du réseau qui gère l'ensemble des abonnés et les services fournis aux abonnés. Ce réseau coeur est responsable de l'établissement de la communication et assure la liaison entre le réseau UMTS et les réseaux extérieurs ;
- le bloc *réseau d'accès* (access-network) : c'est la partie du réseau qui gère l'interface et les ressources allouées sur l'interface radio. Un autre rôle important du réseau d'accès est la gestion de la mobilité de l'usager. La couverture radio étant constituée de cellules de tailles variables, le réseau d'accès doit être capable de faire passer l'usager d'une cellule à une autre en cours de communication. Il s'agit de la fonction de *handover*.

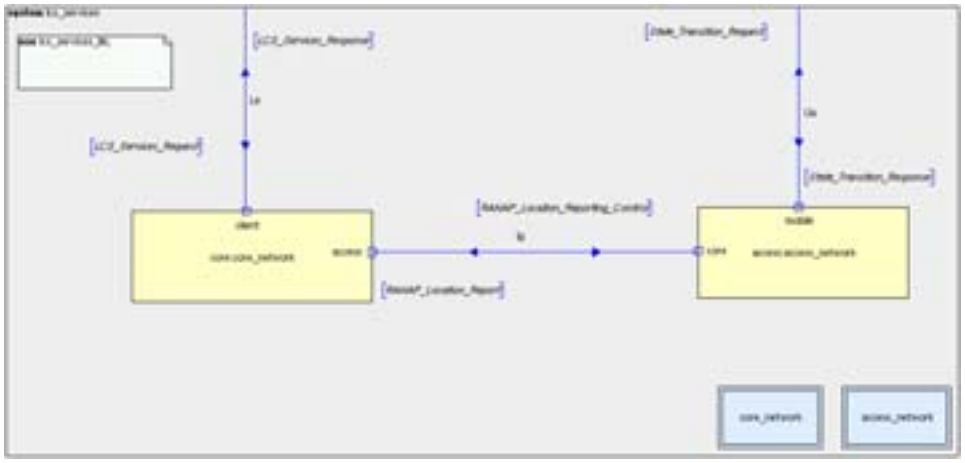


Fig. 4.22 – Système LCS

4.5.2 Spécification du réseau coeur et réseau d'accès (niveau bloc)

Le bloc *réseau coeur* (core-network, voir figure 4.22) est composé des processus GMLC, HLR et MSC.
Ces différents processus décrivent les comportements des entités auxquelles ils s'identifient. Dans ce bloc, le processus GMLC communique avec les processus HLR et MSC via les interfaces Lh et Lg respectivement.
Le bloc *réseau d'accès* (access-network, voir figure 4.24) est composé des processus SRNC et NodeB.
Ces processus décrivent aussi les comportements des entités auxquelles ils se rapportent. Le processus 3G-MSC du bloc réseau coeur communique avec le processus SRNC du réseau d'accès à travers l'interface Iu. Les entités auxquelles ces processus font référence sont détaillées dans la *section 6* du *chapitre 2*.

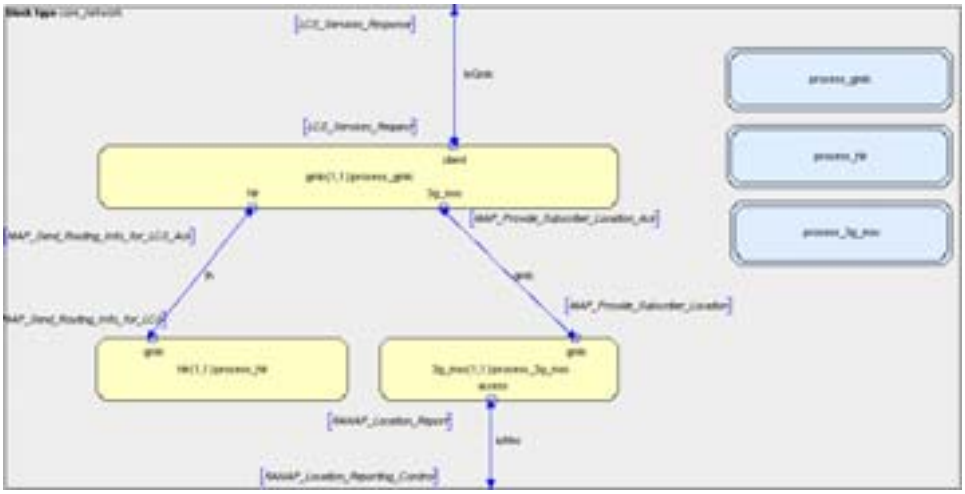


FIG. 4.23 – Bloc réseau de commutation

Les interactions entre l'environnement et les blocs, entre les blocs et entre les différents processus sont matérialisées par les signaux ci dessous. Ceux-ci concernent une requête de localisation en mode commutation de circuit (car utilisation du 3G-MSC et non le SGSN dédié au mode paquet).

1. *lcs-services-request* : signal utilisé par le client LCS externe pour demander une information de localisation d'un terminal au GMLC.
2. *lcs-services-response* : le GMLC retourne l'estimation de la position relative à la requête de localisation
3. *ranap-location-reporting-control* : signal envoyé du 3G-MSC vers le SRNC. Ce message inclut le type d'information de localisation demandée, les capacités du terminal et la QoS.

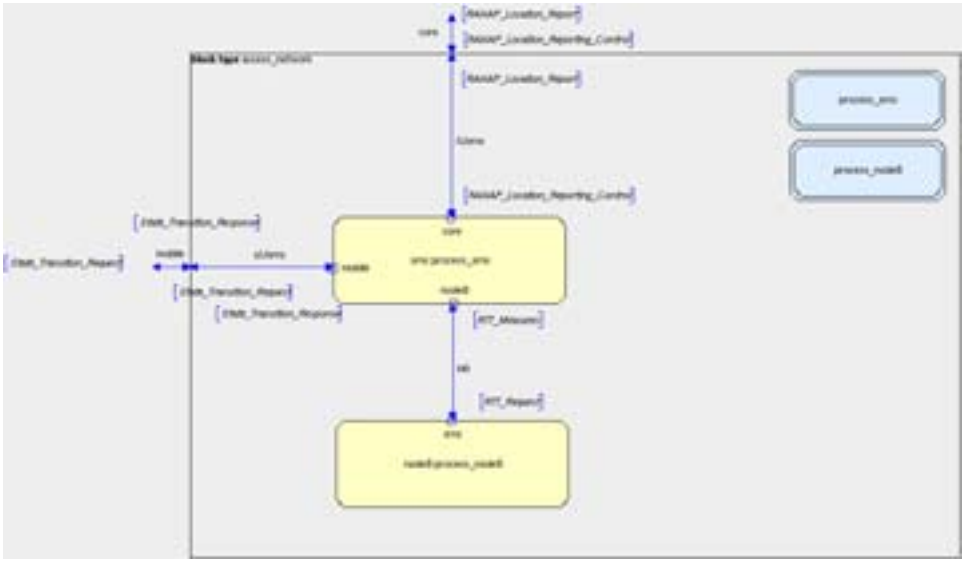


FIG. 4.24 – Bloc réseau d'accès

4. *ranap-location-report* : lorsque la meilleure estimation de la localisation satisfaisant la QoS est obtenue, le SRNC la renvoie au 3G-MSC dans ce signal. Sinon le SRNC renvoie le même signal avec la cause d'échec.
5. *map-send-routing-info-for-lcs* : signal envoyé au HLR par le GMLC en vue de récupérer les informations de routage (localisation du 3G-MSC correspondant) et l'IMSI[19] du terminal recherché.
6. *map-send-routing-info-for-lcs-ack* : le HLR retourne l'adresse courante du 3G-MSC et l'IMSI.
7. *map-provide-subscriber-location* : ce signal transporte le type d'information de localisation demandée (par exemple la localisation courante), l'IMSI de l'abonné, la QoS (temps de réponse, précision), et l'information relative à la priorité du client LCS.
8. *map-provide-subscriber-location-ack* : le 3G-MSC retourne l'information de localisation et son "âge" au GMLC (ceci n'est fait que si la permission a été accordée sinon le 3G-MSC renvoie une réponse d'erreur).
9. *lcs-location-notification-invoke* : pour les applications orientées localisations, le profil d'abonnement nécessite une notification de chaque requête de localisation, ce signal est envoyé au terminal en indiquant le type de requête (exemple la localisation courante), l'identité du LCS client et éventuellement la demande de vérification de l'autorisation d'accéder à l'information de localisation.

[19] International Mobile Subscriber Identity

10. *lcs-location-notification-return-result* : ce signal indique si la permission relative à l'accès aux informations de localisation est accordée ou pas.
11. *state-transition-request* : ce signal effectue un *paging*[20] vers le mobile afin de déterminer l'identifiant de la cellule.
12. *state-transition-response* : réponse relative à la cell-ID (identifiant de la cellule).
13. *rtt-request* : afin d'améliorer la précision, une requête rtt[21] est envoyé au NodeB associé à la cellule.
14. *rtt-measures* : réponse relative à la requête *rtt-request*.

Le processus process_gmlc

C'est le processus par lequel le client LCS accède aux services de localisation. Ce processus demande les informations de routage au HLR. Après avoir vérifié les autorisations relatives au client, il envoie la requête de positionnement au processus 3G-MSC/SGSN et reçoit l'estimation de la localisation de l'entité correspondante via l'interface Lg. Les différents états du processus sont :

- *état null* : après réception de la requête de localisation, le GMLC reste dans *l'état null* pendant que l'identité du client et la nature de la requête de localisation sont vérifiées.
- *état location* : dans cet *état*, le GMLC a envoyé la requête de localisation au 3G-MSC/SGSN et attend la réponse contenant la position estimée du mobile. Le processus GMLC active un *timer* dans cet état afin de limiter le temps d'attente de la réponse du processus 3G-MSC. Si le *timer* expire avant la réception de la réponse, le GMLC indique un échec de localisation au client et passe dans l'état *null*.
- *état interrogation* : dans cet *état*, le GMLC a envoyé une interrogation au HLR correspondant au terminal à localiser et est en attente de la réponse donnant l'adresse du 3G-MSC/SGSN et l'IMSI du terminal. Le processus GMLC active un *timer* dans cet *état* afin de limiter le temps d'attente de la réponse du processus HLR. Si le *timer* expire avant la réception de la réponse, le GMLC indique un échec de localisation au client et passe dans l'état *null*.

Le processus process_hlr

Le processus HLR contient les abonnements aux services LCS et les informations de routage. Il est accessible par le processus GMLC via l'interface Lh. Un seul état caractérise ce processus :

[20] le *paging* est une opération qui consiste à rechercher un abonné

[21] Round Trip Time : intervalle de temps écoulé entre le moment où le signal est envoyé et le moment où il est reçu

– *état null* : dans cet état, le processus HLR est inactif.

Le processus process_3g_msc

Le processus 3G-MSC contient les fonctionalités relatives à l'autorisation d'abonnement et gère les requêtes de localisation. Les différents états du processus sont :

– *état idle* : dans cet *état*, le processus 3G-MSC est inactif.
– *état location* : dans cet *état*, le processus attend une réponse du réseau d'accès et active un *timer* en vue de limiter le temps d'attente. Si le *timer* expire avant la réception de la position du mobile, le 3G-MSC indique un échec de la localisation au processus demandeur et repasse dans l'état *idle*.

Le processus process_srnc

Ce processus a une fonction de routage des communications entre le NodeB et le réseau coeur d'une part et d'autre part le contrôle et la supervision du NodeB. Il contrôle le flux des requêtes de localisation. Il a aussi un rôle de sélection des méthodes de localisation et de calcul des positions.
Les différents états du processus sont :

– *état idle* : le processus est inactif dans cet *état*.
– *état rtt* : dans cet *état*, le processus est en attente de la réponse *RTT*.

Le processus process_nodeB

Ce processus fournit des résultats de mesure en rapport avec l'estimation de la position des terminaux et mesure les signaux radio et les communique au SRNC. Un état caractérise ce processus :

– *état idle* : dans cet *état* le processus est inactif.

4.5.3 Vérification et Validation du modèle

Paramètres du simulateur

– *environnement du système* : le fichier *feeds*[22] est constitué des signaux suivants :

```
uu state-transition-response srnc
le lcs-services-response gmlc
paging lcs-location-notification-return-result 3g-msc
```

FIG. 4.25 – Lignes de feed

[22]liste des messages stimuli envoyés par le testeur vers le système sous test

– *réduction de la taille d'espace des états* : nous avons utilisé la même technique que pour la couche WTP pour réduire la taille d'espace des états.

Simulation interactive

Pour exercer le modèle, différentes simulations simples ont été exécutées :
– *scénario 1* : ce scénario représente le succès d'une requête de localisation (figure 4.26) d'un abonné par le client LCS. Dans ce scénario, une requête est lancée par un client externe, toutes les interactions sont effectuées jusqu'à obtention du résultat (la position du mobile à localiser) ;
– *scénario 2 et 3* : dans la norme, nous observons que des *timers* sont activés dans les états location et interrogation du GMLC (figure 4.27), puis dans l'état location du 3G_MSC (figure 4.28). Il est donc important d'identifier les problèmes relatifs aux expirations des timers. Pour cela, nous proposons des scénarios d'échec relatifs aux expirations de ces derniers. Ainsi :
  • dans la figure 4.27, le timer qu'active le processus GMLC lorsqu'il transmet la requête de localisation (*map-provide-subscriber-location*) au 3G-MSC expire avant la réception du résultat. Le GMLC renvoie alors au client une réponse d'échec ;
  • dans figure 4.28, le timer qu'active le processus 3G-MSC lorsqu'il transmet la requête de localisation au réseau d'accès (et plus particulièrement au SRNC) arrive à expiration et une réponse d'échec est retransmise au GMLC puis au client LCS.

Vérification des propriétés

Les propriétés suivantes ont été vérifiées :
– propriété 1 : *succès d'une localisation globale*, cette propriété représente la satisfaction d'une requête de localisation dans sa globalité ;
– propriété 2 : *profil de l'abonné*, pour les applications à valeur ajoutée (exemple application orientée localisation), le profil d'abonnement nécessite une notification au terminal de chaque requête de localisation (voir les interactions 9 et 10 de la section 4.5.1) pour la réussite d'une requête.
Ces deux proprétés sont représentées par les MSC de la figure 4.29.

– Options de la simulation exhaustive :
Nous limitons la profondeur de notre simulation à 35. Cette longueur est comparée avec la longueur nominale du scénario relatifs à la *propriété 1.* Dans notre cas il est égal à 17, incluant les 5 étapes initiales. Notre limite correspond alors au double de la longueur nominale.

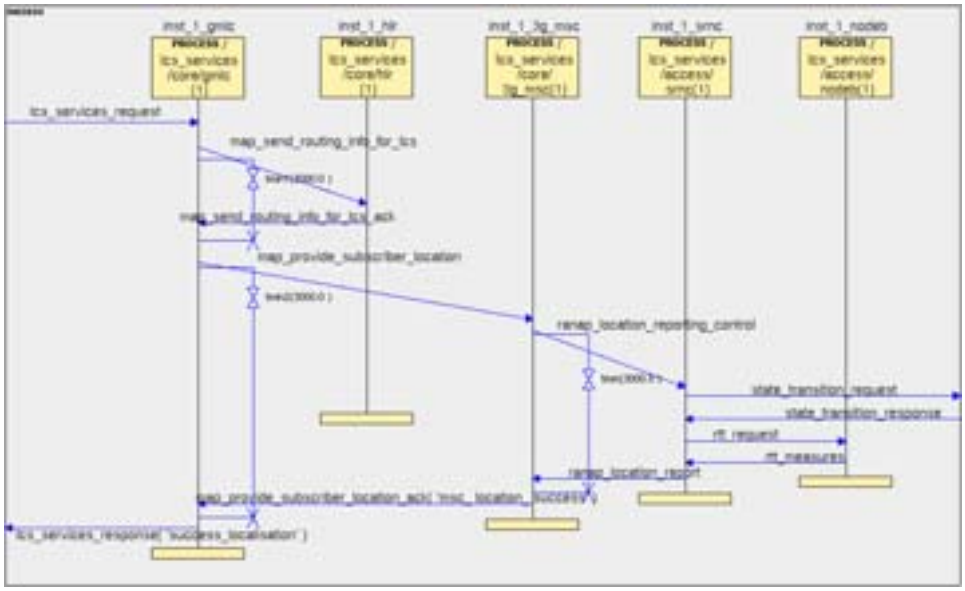


FIG. 4.26 – Requête de localisation réussie

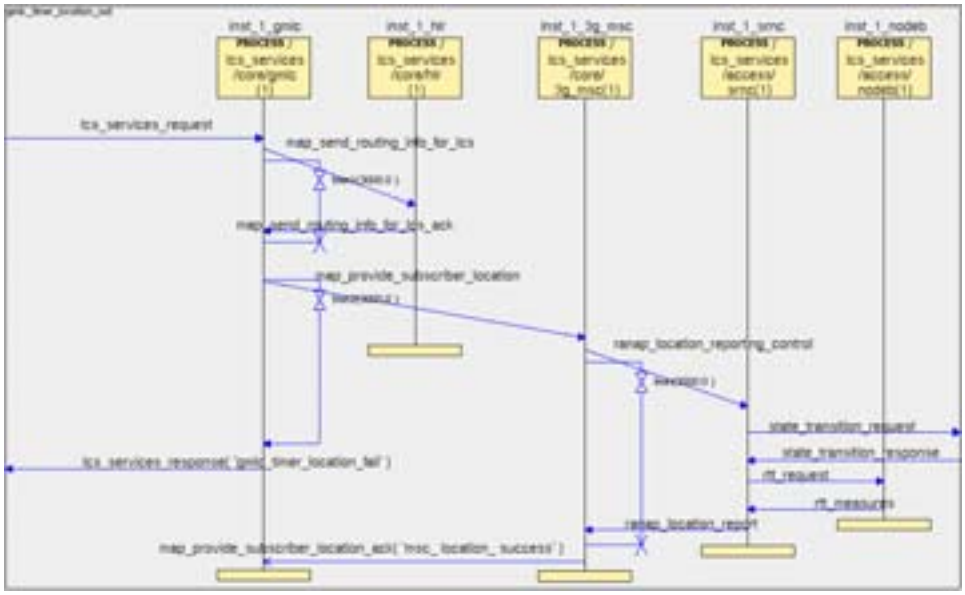


FIG. 4.27 – Requête impliquant une expiration du timer du GMLC

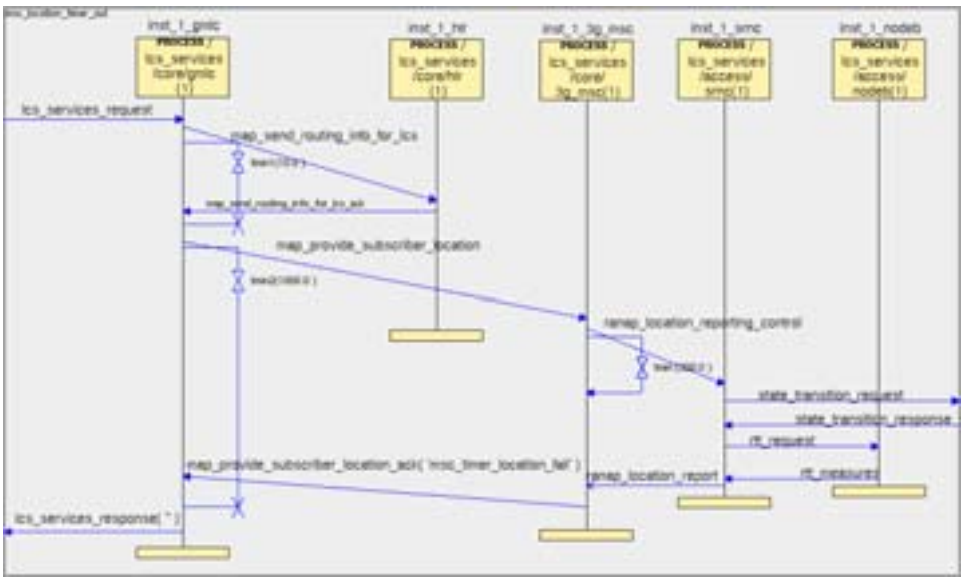


FIG. 4.28 – Requête impliquant une expiration du timer du MSC

Le fichier suivant décrit les options de la simulation :

```
mode breadth
define edges_dump ''
define states_dump ''
deadlock limit 5
exception limit 5
stop limit 5
define stop_cut true
error limit ot_success_localisation 1
success limit ot_success_localisation 1
error cut ot_success_localisation = true
success cut ot_success_localisation = false
define states_limit 0
define depth_limit 100
define depth_limit_stop false
define hash_fill_limit 10
define main_hash_size 100000
define hash_size 1000
define verify_stats false
```

Compte tenu des options décrites précédemment et des propriétés à vérifier, la simulation exhaustive de notre modèle a donné les principaux résultats suivants :

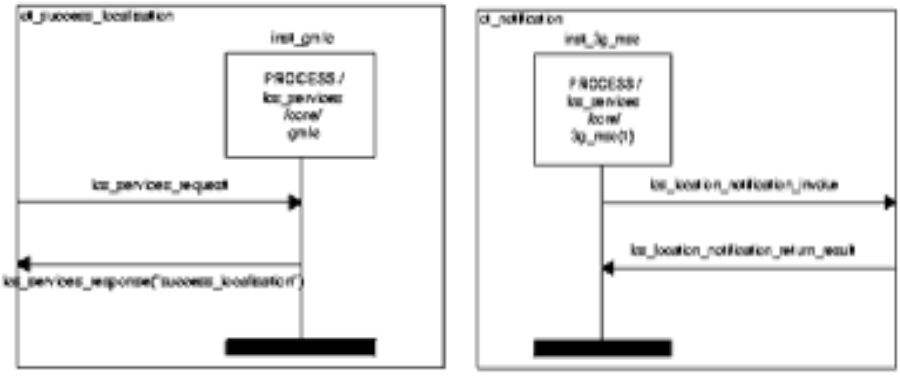


FIG. 4.29 – Succès d'une localisation globale (gauche) et profil de l'abonné (droite)

– *propriété 1* :
  - *graphe d'état* : environs 52270 états (nombre de différents états atteints), 138381 transitions (nombre de transitions consommées, soit une moyenne d'environs 2,64 transitions par état), avec une profondeur (profondeur maximale atteinte lors de la recherche) de 100 transitions consécutives et une largeur maximale de 1514 états ;
  - *performance* : ce graphe d'états a été généré en 17 s, soit plus de 8140 transitions consommées à la seconde, ce qui correspond à environs 29000000 de transitions par heure (pour une machine Unix de 2 Go de RAM et de CPU 1133 Mhz) ;
  - 85,71% de transitions et 100% des états ont été respectivement consommées et atteints ;
  - 5939 succès ont été détectés, en d'autres termes, aucune violation de la propriété n'a été décelée ;
  - aucune exception et aucun deadlock n'ont été décelés, généralement (mais pas nécessairement) ils correspondent à des erreurs de conception ;
  - la simulation s'est arrêtée après que la profondeur maximale ait été atteinte. Le graphe d'état ayant presque été en totalité exploré.

– *propriété 2* : les résultats relatifs à cette propriété sont équivalents à ceux que nous avons commentés pour la propriété précédente.

Nos statistiques sont satisfaisantes car plus de 85% de transitions et 100% des états ont été respectivement consommées et atteints pour les deux

```
(8192 states 20894 transitions 3 seconds, depth=52, breadth=454)
(16384 states 42519 transitions 6 seconds, depth=66, breadth=700)
(24576 states 64287 transitions 9 seconds, depth=76, breadth=917)
(32768 states 86276 transitions 11 seconds, depth=84, breadth=1049)
(40960 states 108041 transitions 14 seconds, depth=91, breadth=1249)
(49152 states 130085 transitions 16 seconds, depth=97, breadth=1350)
verify stopped by depth limit

Number of states : 52270
Number of transitions : 138381
Maximum depth reached : 100
Maximum breadth reached : 1514
duration : 0 mn 17 s

Number of exceptions : 0
Number of deadlocks : 0
Number of stop conditions : 0
Transitions coverage rate : 85.71 (4 transitions not covered)
States coverage rate : 100.00 (0 states not covered)
Basic blocks coverage rate : 92.31 (2 basic blocks not covered)
Number of errors : 0
Number of success : 5939
```

FIG. 4.30 – Propriété 1 : Succès d'une localisation globale

propriétés. Aucune violation de nos propriétés, aucun deadlock ni exception n'ont été décelés. En plus d'une preuve générale de correction (comme l'absence de blocage), la cohérence de la spécification SDL avec le comportement souhaité du système a été vérifiée pour les services de localisation.
Un certain nombre de problèmes et/ou d'ambiguïtés ont été observés dans les spécifications, cela pourrait être à l'origine de problèmes d'intéropérabilité futurs. Par exemple, lors de l'expiration du timer de localisation au niveau du processus GMLC (figure 4.27), il serait intéressant de re-utiliser le résultat renvoyé (quoi que en retard) par le processus MSC lors d'une requête future si celle-ci se fait dans un délai "raisonnable". Dans le cas présent, la norme ne précise rien. Il en est de même lors de l'expiration du timer du processus MSC (figure 4.28).
Par ailleurs, dans les deux cas précédents, la norme parle vaguement de réponse d'échec alors qu'il serait intéressant d'être plus précis afin qu'on sache à tout moment localiser le problème qui serait à l'origine d'un échec de la localisation. Dans notre spécification, nous avons noté : *gmlc_timer_location_fail* en paramètre au message d'erreur qui résulterait d'une expiration du timer côté processus GMLC lors de la localisation et, *msc_timer_loca-*

```
(8192 states 23413 transitions 16 seconds, depth=45, breadth=572)
(16384 states 48133 transitions 18 seconds, depth=56, breadth=817)
(24576 states 73205 transitions 21 seconds, depth=65, breadth=1127)
(32768 states 98453 transitions 24 seconds, depth=71, breadth=1297)
(40960 states 123742 transitions 27 seconds, depth=77, breadth=1560)
(49152 states 149318 transitions 29 seconds, depth=82, breadth=1725)
(57344 states 175057 transitions 32 seconds, depth=86, breadth=1831)
(65536 states 200665 transitions 35 seconds, depth=90, breadth=2037)
(73728 states 226383 transitions 38 seconds, depth=94, breadth=2208)
(81920 states 252156 transitions 40 seconds, depth=98, breadth=2358)
verify stopped by depth limit

Number of states : 86259
Number of transitions : 265852
Maximum depth reached : 100
Maximum breadth reached : 2529
duration : 0 mn 42 s

Number of exceptions : 0
Number of deadlocks : 0
Number of stop conditions : 0
Transitions coverage rate : 85.71 (4 transitions not covered)
States coverage rate : 100.00 (0 states not covered)
Basic blocks coverage rate : 92.31 (2 basic blocks not covered)
Number of errors : 0
Number of success : 27604
```

FIG. 4.31 – Propriété 2 : Profil de l'abonné

tion_fail dans le cas du processus MSC.

4.6 Conclusion

Ce chapitre a poursuivi l'étude et le détail de la stratégie de test présentée dans le chapitre 3. Nous avons tout d'abord présenté une méthodologie de génération de test pour la couche applicative WAE et en particulier pour les applications écrites en langage WML. C'est une méthode de reverse engineering car nous partons du code WML vers le modèle SDL. Cette méthodologie passe d'abord par la proposition d'une architecture de test, puis la génération de l'arbre de comportement de l'application WML, et enfin la transformation de l'arbre obtenu en modèle SDL sur lequel seront appliquées les techniques classiques de génération de tests.
Nous avons ensuite appliqué la stratégie de test développée pour vérifier et

valider les protocoles WSP et WTP puis les services de localisation dans un réseau UMTS. Pour ce faire, nous avons dans un premier temps modélisé ces protocoles et services dans le formalisme SDL puis nous avons utilisé l'outil ObjectGeode dans les trois modes de simulation (aléatoire, interactive, exhaustive) afin de détecter les erreurs de conception et vérifier le comportement des modèles par rapport à certaines propriétés.
De façon générale, les taux de couverture des transitions/états ont été satisfaisants, nous n'avons pas décelé de situations de blocage, et avons vérifié sans problème les propriétés souhaitées. Cependant, nous avons relevé plusieurs ambiguïtés sur nos modèles (WSP, WTP et services de localisation) qui pourraient être à l'origine des problèmes futurs d'intéropérabilité. Nous avons donc souvent dû interpréter la norme [28, 29]. Les scénarios engendrés, représentés sous le format MSC [39] seront utilisés sur une plate-forme réelle.

Chapitre 5

Validation de l'intéropérabilité par contrôle et observation des fichiers LOG (VaICOLOG)

5.1 Introduction

Ce chapitre a comme objectif l'expérimentation des méthodes de test d'intéropérabilité sur des protocoles et services liés à la mobilité. La mise en oeuvre de tests proposés s'est portée sur les protocoles WSP et WTP.

Ces tests doivent permettre de vérifier l'interfonctionnement entre une application sur un terminal et une application sur un serveur. Cette vérification implique l'observation des interactions provoquées par une requête du terminal. Il s'agit donc d'un test d'intéropérabilité des services au sein de l'architecture de communication globale.
La section suivante présente l'architecture relative aux contrôles et aux observations nécessaires au test d'interopérabilité. La section 3 présente notre méthodologie de vérification de l'intéropérabilité par la génération des fichiers LOG observés (PO) au niveau de la passerelle, le contrôle des requêtes émises (PCO) à partir d'un PDA et par l'observation du réseau grâce au snifer ethereal.

5.2 Architecture de test d'intéropérabilité des protocoles et services

Dans le cas des systèmes que nous étudions, les implantations à tester n'offrent pas d'interfaces directes (c'est le cas des couches protocolaires du

WAP). Cela impose donc de définir des PCOs[1] et des POs[2].
Pour pouvoir analyser les échanges de données, il est nécessaire d'observer le transit en différents points stratégiques. Ceci conduit à l'installation de PO et PCO comme suit :

- la passerelle étant une entité essentielle du dispositif de l'architecture du WAP, on souhaite observer si celle-ci traite les données (entre la réception et l'émission de celles-ci) conformément à la norme [28, 29] et plus précisément au niveau des couches WSP et WTP. Ce qui nous a donc conduit à installer un PO entre la couche WSP et la couche WTP puis entre la couche WTP et la couche WDP ;
- un autre PO entre la passerelle et le serveur permet d'observer si les données sont bien retranscrites lors du transfert de la passerelle au serveur et vice-versa ;
- le PCO situé à l'amont du terminal permet d'initialiser les transactions, d'injecter les évènements et de récupérer les résultats.

Nous avons des contrôles et observations à quatre niveaux. Le pouvoir de détection des erreurs est assez fort car nous capturons toutes les données échangées entre le terminal et la passerelle, entre la passerelle et le serveur puis au niveau de la passerelle entre les couches WSP et WTP puis WTP et WDP. On peut effectuer des test sans PO au départ, et s'ils échouent, rajouter progressivement les PO et diagnostiquer où l'erreur s'est produite, ce qui introduit une méthodes incrémentale basée sur le pouvoir de test. Les architectures de test présentées ci-dessous (figures 5.1 et 5.2) ne sont que l'implantation directe du modèle décrit dans la section 3.4.2 (voir figure 3.7).

5.2.1 Architecture globale

Nous retrouvons dans cette architecture (figure 5.1) les différents PO et PCO qui permettent de récupérer les différents verdicts locaux. Le protocole WAP utilise une passerelle qui permet de transformer les données provenant d'un navigateur WAP (dans le format WML) en données compatibles avec les serveurs WEB (dans le format HTTP). Cette passerelle implante donc les différentes couches protocolaires nécessaires à son bon fonctionnement (WDP/UDP, WTP, WSP, etc..).

5.2.2 Architecture en couche

La figure 5.2 présente une architecture du point de vue des couches protocolaires de la passerelle, du terminal et des équipements intermédiaires. Est aussi présentée, la situation des différents PCO et PO implémentés à travers l'architecture :

[1] Point de Contrôle et d'Observation
[2] Point d'Observation

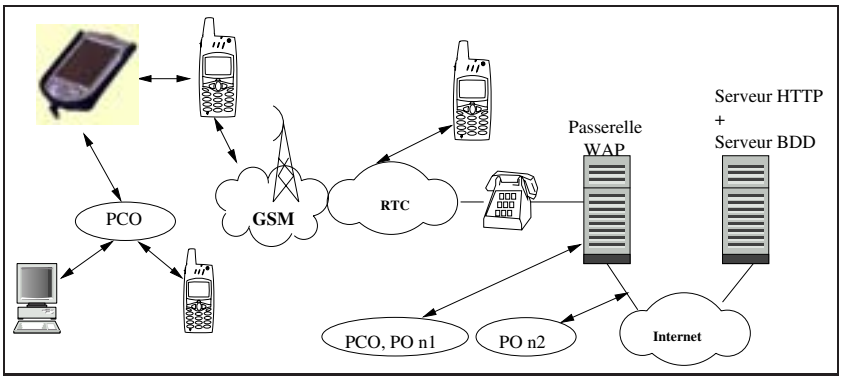


FIG. 5.1 – Architecture de test pour le WAP

– WSP_Layer_PO.log : c'est le fichier LOG relatif au PO implanté au niveau de la couche WSP côté passerelle WAP,
– WTP_Layer_PO.log : c'est le fichier LOG relatif au PO implanté au niveau de la couche WTP côté passerelle WAP,
– WSP_PO.log : c'est le fichier LOG relatif au PCO implanté au niveau de la couche WSP côté PDA/simulateur.

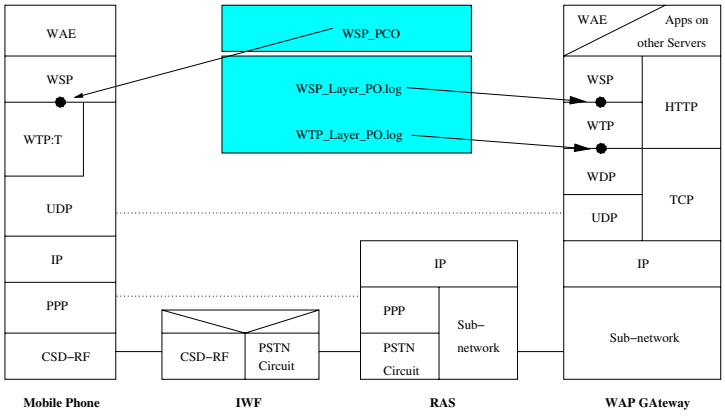


FIG. 5.2 – Les PCOs et POs dans l'architecture

5.3 Validation de l'interopérabilité par Contrôle et Observation des fichiers LOG (VaICOLOG)

Dans cette section, nous présentons l'installation pratique des différents POs et/ou PCOs (figure 5.1) qui vont former l'architecture de test. Dans ce qui suit, chaque PO est constitué de deux parties, l'une *PO_trace* est dédiée à la scrutation du réseau et l'autre *PO_analyse* est chargé de fournir

le verdict local.

5.3.1 Le PO n°1 : Observations au niveau de la passerelle

Le PO n°1 observe le trafic reçu et émis par la passerelle WAP. La passerelle Kannel étant un logiciel libre, nous disposons ainsi des codes sources et nous avons pu les modifier pour installer des outils de traces. L'architecture logicielle de Kannel est structurée en couches implémentées par des threads et qui communiquent entre elles par échange de message. Du fait de cette implémentation, nous avons mis en place des POs entre ces différentes couches. Le PO installé est constitué d'un *PO_trace_in* qui récupère le trafic entrant, d'un *PO_trace_out* qui récupère le trafic sortant et d'un *PO_analyse* qui exploite l'ensemble des traces pour fournir le verdict local.

Ces POs peuvent être utilisés lorsque nous n'avons pas la possibilité de programmer des PCOs au niveau de la passerelle, ainsi nous pouvons tester le comportement de la passerelle. Les informations récupérées au niveau de la passerelle sont assez descriptives, on y retrouve les noms des primitives, les données transférées, etc..

Observations dans le cas d'une requête réussie

Ce cas implémente un scénario vu dans le chapitre précédent. C'est le scénario d'une requête émise par un terminal qui reçoit par suite le résultat à sa demande, c'est le cas idéal. Nous allons à travers ces observations vérifier que la passerelle *KANNEL* implémente bien la norme à partir des résultats suivants :

- la figure 5.3 montre un morceau de trace d'exécution obtenu à partir du PO de la couche WSP. Cette couche permet d'établir et de relâcher une session entre un client et un serveur selon les deux modes de connexion présentés précédemment,
- la figure 5.4 montre un morceau de trace d'exécution obtenu à partir du PO de la couche WTP. Cette couche implémente le protocole de transport qui s'appuie sur un service de datagramme.

Dans les deux cas, il y a bien cohérence entre les observations effectuées sur l'implantation (partie gauche des figures suivantes) et la norme (partie droite des figures suivantes).

Observations dans le cas d'une requête erronée

Dans ce cas, nous avons émis une requête et le terminal nous a renvoyé un message d'erreur non explicite. N'ayant pas obtenu de résultat relatif à notre requête, l'architecture mise en place nous permet donc d'observer au niveau des couches WSP (figure 5.5) et WTP (figure 5.6) ce qu'il en est :

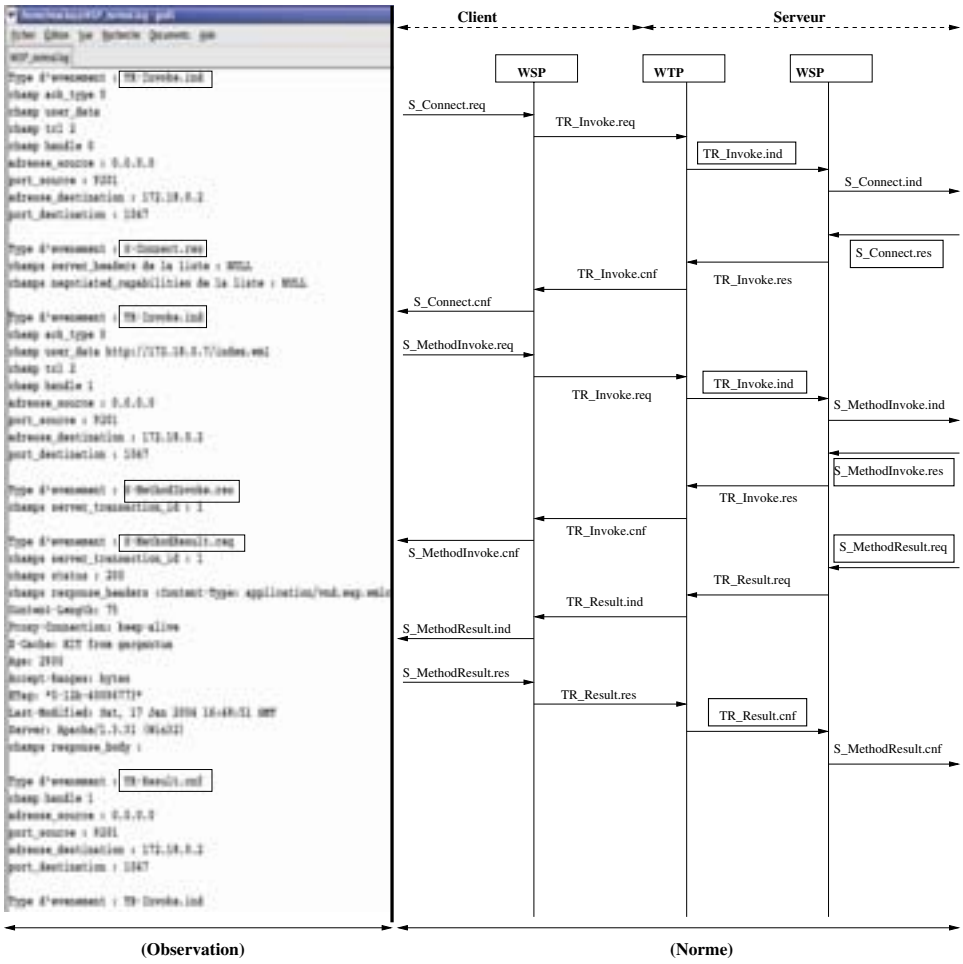


FIG. 5.3 – Fichier Log relatif à l'observation de WSP

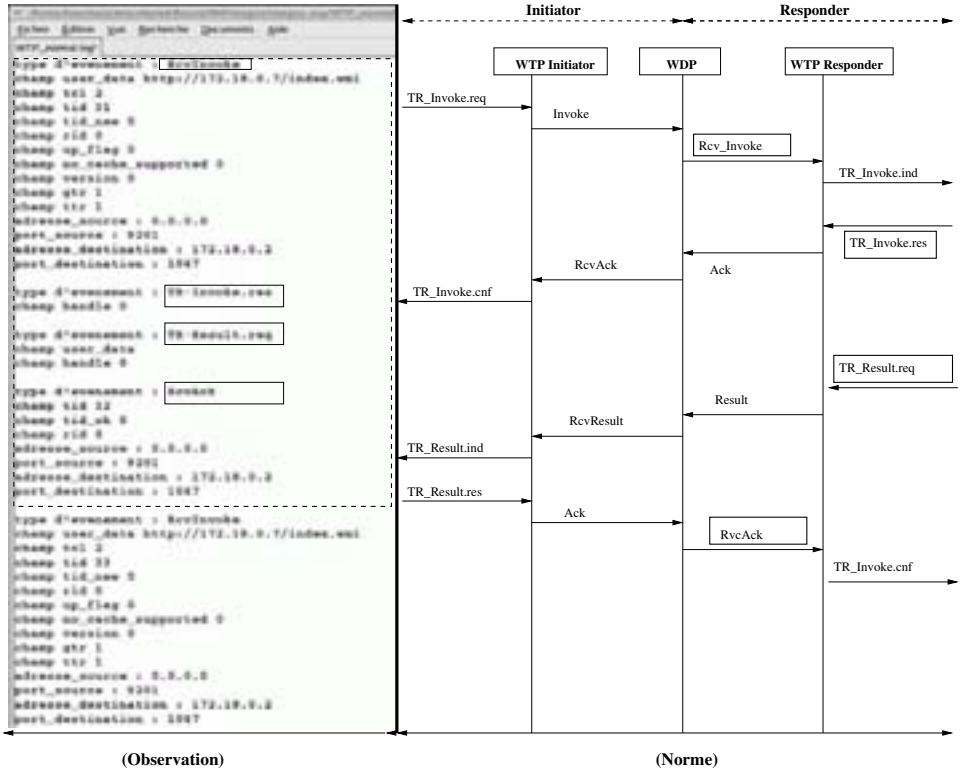


FIG. 5.4 – Fichier Log relatif à l'observation de WTP

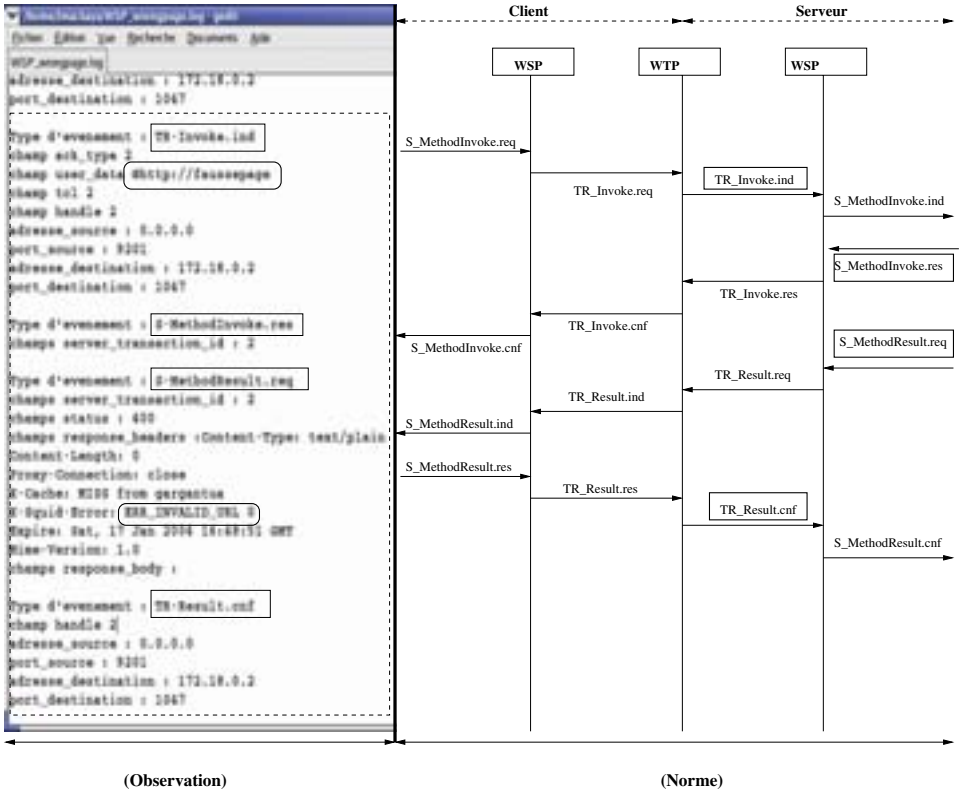


FIG. 5.5 – Fichier Log relatif à l'observation de WSP avec demande de page erronée

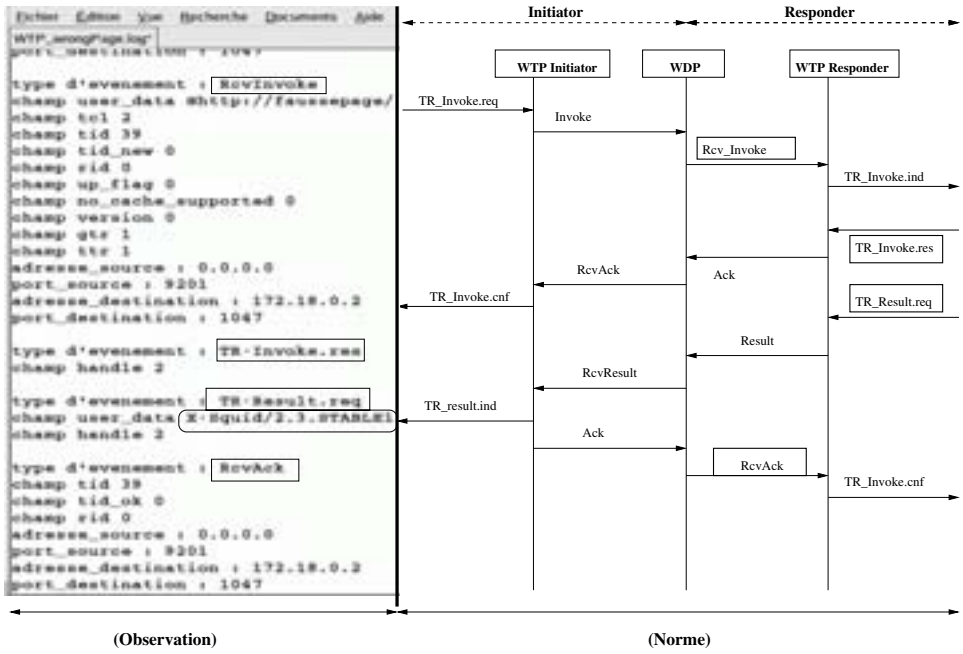


FIG. 5.6 – Fichier Log relatif à l'observation de WTP avec demande de page erronée

- couche WSP : après une connexion réussie, il apparaît le champ *X-Squid-Error* dans la primitive de service *S-MethodResult.req* avec la valeur *ERR_INVALID_URL* qui nous indique clairement que l'URL demandée est invalide,
- couche WTP : ici aussi on observe la présence de la variable *X-Squid* dans le champ *user_data* de la réponse à la requête de transaction *TR_Result.req*.

Il convient quand même de signaler ici que la norme n'est pas très bavarde sur les valeurs renvoyées par le champ *response_headers* de la primitive *S-MethodResult.req* (primitive utilisée pour fournir une réponse à une requête préalable) notamment en ce qui concerne la valeur du champ *X-Squid-Error*. On peut supposer que l'implémentation s'en charge. Dans le cas de KANNEL, il y a la valeur *ERR_INVALID_URL* affectée à la variable *X-Squid* pour implémenter une *url* invalide.

Amélioration de l'interface graphique

La figure 5.7 est une interface graphique développée en vue du lancement de kannel (lancement de la Bear box et Wap box). En effet, lancer *KANNEL* nécessite l'ouverture d'au moins deux fenêtres de lignes de commande (une

pour le Bear box[3] et l'autre pour la Wap box[4]). Cette interface est donc une amélioration apportée à *KANNEL* dans le cadre de ce projet en vue de la rendre plus facile d'utilisation. Ainsi, nous avons :

- le bouton *bearerbox* pour lancer la Bear box,
- le bouton *wapbox* pour lancer la Wap box,
- l'ensemble de bouton à cocher (WSP, WTP, et WTLS, WDP en prévision) pour choisir la couche pour laquelle on souhaite générer un fichier *log*,
- le bouton *valid* pour valider la couche choisie et générer le fichier *log*,
- le bouton *quit* pour arrêter *KANNEL*.

FIG. 5.7 – Interface graphique de kannel

5.3.2 Le PO n°2 : Observation par le snifer

Les passerelles WAP sont en général reliées à internet via des réseaux locaux. Or les données échangées sur de tels réseaux peuvent être recueillies par toute machine se trouvant sur le réseau. Il est donc judicieux d'utiliser pour le PO n°2 un snifeur comme élément *PO_trace* (ce qui ne gêne en rien le trafic de données). Pour ce faire, nous avons utilisé *ethereal* (figure 5.8). C'est un outil graphique de capture et d'analyse des trames circulant sur le réseau. Le *PO_analyse* dispose d'un ensemble de propriétés à vérifier. Il

[3]ce processus se connecte au centre des SMS et aux routeurs CSD (Circuit Switched Data) et constitue une interface unique entre ces derniers et les autres processus de la passerelle. La Bearer box implémente la couche WDP du protocole WAP

[4]ces processus implémentent les couches hautes du protocole WAP (WSP, WTP, WTLS). Chaque session et toutes les transactions utilisées par cette dernière sont gérées par la même WAP box. Les sessions et transactions ne sont pas échangées entre WAP box.

vérifie ces propriétés sur les traces fournies par *PO_trace*. Une fois le test fini, il envoie son verdict local au testeur chargé de centraliser les verdicts locaux.

Comme on peut l'observer sur la figure 5.8, l'interface graphique est divisée en trois parties :

- la première partie est du type général, on y trouve des informations de type adresse IP des machines sources et destinataires ou encore les protocoles utilisés lors de l'échange des données,
- la deuxième partie détaille la trame sélectionnée,
- la troisième partie est une vision de la trame en codage hexadécimal.

5.3.3 Le PCO côté PDA

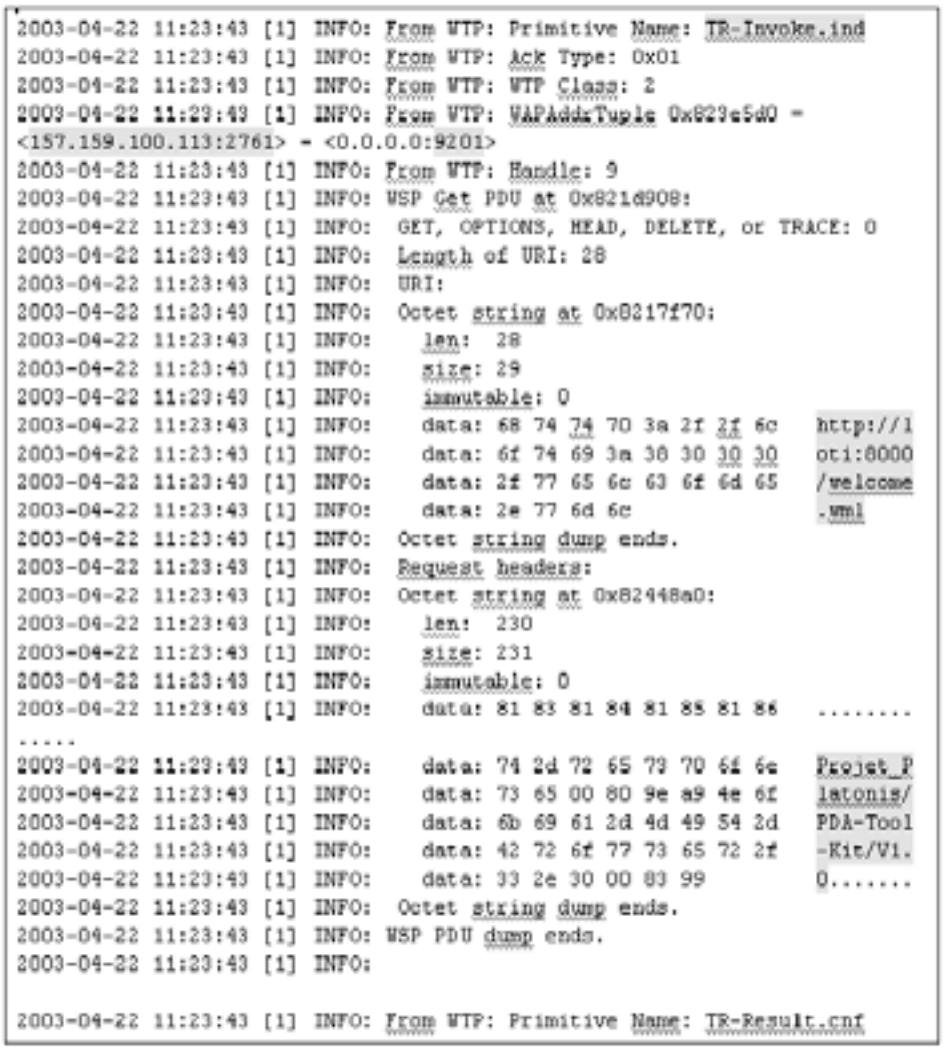
Dans la figure 5.9, la connexion entre le client (programme implantant le PCO) et le serveur (avec l'adresse IP 157.159.100.113) a été établie en utilisant le mode de connexion (9201, mode orienté connexion). Le message utilisé pour ouvrir la page (welcome.wml) est le message *TR-Invoke.ind*. Le PCO doit être capable d'envoyer et de recevoir différentes trames, ainsi que les différents verdicts locaux. Nous avons choisi d'utiliser un PDA (avec un système d'exploitation Windows CE) car il est assez facilement programmable et il peut communiquer soit directement en GSM, soit par l'intermédiaire d'un téléphone mobile équipé d'un port Irda. Nous avons redéveloppé un navigateur WAP (en implantant les couches WTP et WSP suivant la norme WAP 1.1) et en mettant en place une interface graphique permettant de charger les fichiers de test.

Durant le déroulement d'un test, le PDA attend des données qui peuvent être soit un retour d'informations, soit un verdict local. A la fin du test, le PDA attend d'avoir tous les verdicts locaux pour donner le verdict final suivant la règle énoncée précédemment.
La figure 5.10, montre le début d'un test, avec la demande de connexion (même communication que pour la figure 5.9 mais avec une autre vue). Dans ce qui suit, nous détaillons un exemple de scénario qui permet de tester une transaction de classe 2 avec un message abort du répondeur. Le scénario ainsi produit est injecté au niveau du PCO, et nous pouvons observer les échanges au niveau du PO (voir figure 5.11). Ainsi, nous sommes en mesure d'observer que le mobile s'est correctement comporté, suite à l'envoi d'un message inopportun au moment de l'échange. Comme résultat de cette erreur injectée, la page wml à laquelle le terminal souhaitait accéder ne peut être ouverte en raison du message envoyé via le PCO.

5.3.4 Le PO avec le simulateur WesternWapper

WesternWapper est un navigateur développé par [54]. Il est écrit en java et implémente les couches WSP et WTP du protocole WAP. Cette ap-

FIG. 5.8 – Capture et Analyse de trames par Ethereal



```
2003-04-22 11:23:43 [1] INFO: From WTP: Primitive Name: TR-Invoke.ind
2003-04-22 11:23:43 [1] INFO: From WTP: Ack Type: 0x01
2003-04-22 11:23:43 [1] INFO: From WTP: WTP Class: 2
2003-04-22 11:23:43 [1] INFO: From WTP: WAPAddrTuple 0x829e540 -
<157.159.100.113:2761> - <0.0.0.0:9201>
2003-04-22 11:23:43 [1] INFO: From WTP: Handle: 9
2003-04-22 11:23:43 [1] INFO: WSP Get PDU at 0x821d908:
2003-04-22 11:23:43 [1] INFO:  GET, OPTIONS, HEAD, DELETE, or TRACE: 0
2003-04-22 11:23:43 [1] INFO:  Length of URI: 28
2003-04-22 11:23:43 [1] INFO:  URI:
2003-04-22 11:23:43 [1] INFO:  Octet string at 0x8217f70:
2003-04-22 11:23:43 [1] INFO:    len:  28
2003-04-22 11:23:43 [1] INFO:    size: 29
2003-04-22 11:23:43 [1] INFO:    immutable: 0
2003-04-22 11:23:43 [1] INFO:    data: 68 74 74 70 3a 2f 2f 6c   http://l
2003-04-22 11:23:43 [1] INFO:    data: 6f 74 69 3a 38 30 30 30   oti:8000
2003-04-22 11:23:43 [1] INFO:    data: 2f 77 65 6c 63 6f 6d 65   /welcome
2003-04-22 11:23:43 [1] INFO:    data: 2e 77 6d 6c               .wml
2003-04-22 11:23:43 [1] INFO:  Octet string dump ends.
2003-04-22 11:23:43 [1] INFO:  Request headers:
2003-04-22 11:23:43 [1] INFO:  Octet string at 0x82448a0:
2003-04-22 11:23:43 [1] INFO:    len:  230
2003-04-22 11:23:43 [1] INFO:    size: 231
2003-04-22 11:23:43 [1] INFO:    immutable: 0
2003-04-22 11:23:43 [1] INFO:    data: 81 83 81 84 81 85 81 86   ........
.....
2003-04-22 11:23:43 [1] INFO:    data: 74 2d 72 65 73 70 6f 6e   Projet_P
2003-04-22 11:23:43 [1] INFO:    data: 73 65 00 80 9e a9 4e 6f   latonis/
2003-04-22 11:23:43 [1] INFO:    data: 6b 69 61 2d 4d 49 54 2d   PDA-Tool
2003-04-22 11:23:43 [1] INFO:    data: 42 72 6f 77 73 65 72 2f   -Kit/V1.
2003-04-22 11:23:43 [1] INFO:    data: 33 2e 30 00 83 99         0.......
2003-04-22 11:23:43 [1] INFO:  Octet string dump ends.
2003-04-22 11:23:43 [1] INFO: WSP PDU dump ends.
2003-04-22 11:23:43 [1] INFO:

2003-04-22 11:23:43 [1] INFO: From WTP: Primitive Name: TR-Result.cnf
```

FIG. 5.9 – WSP PO dans kannel

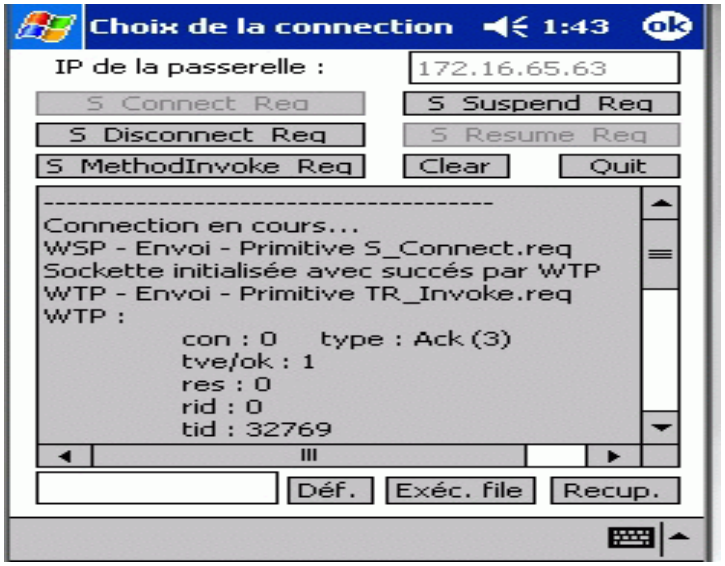


FIG. 5.10 – WSP PCO côté PDA

plication a l'avantage de fournir un fichier log (figure 5.12, partie (a)) du client WSP qui peut donc être analysé dans le cadre de nos observations. Un exemple de ligne dans le fichier de log fourni par *WesternWapper* est *"wtp_send invoke_pdu 1"*, qui indique que la couche WPT a envoyé un message d'invocation *"invoke"* avec 1 comme identification de transaction (transaction ID). La partie (b) de la figure présente un diagramme des séquences relatives aux traces d'exécution du fichier log (a). Ce diagramme montre que la norme est bien respectée.
Cette application contient plusieurs modules et se lançait à partir d'une ligne de commande dans laquelle il fallait indiquer au module de lancement l'adresse IP de la passerelle et l'URL visée. Pour la rendre plus conviviale et plus facile d'utilisation, nous y avons ajouté une interface (figure 5.12, partie (c)) qui s'utilise intuitivement :

- boîte de dialogue *WAP page* : pour renseigner l'URL désirée,
- boîte de dialogue *gateway* : pour renseigner l'adresse IP de la passerelle,
- le bouton *valid* : pour valider les deux points précédents et lancer la requête,
- le bouton *display* : pour l'affichage du fichier log,
- le bouton *quit* : pour quitter l'application.

5.4 Conclusion

Plusieurs techniques sont utilisées dans le cadre de la vérification et validation des protocoles. Nous avons ici exposé notre méthode VaCOLOG

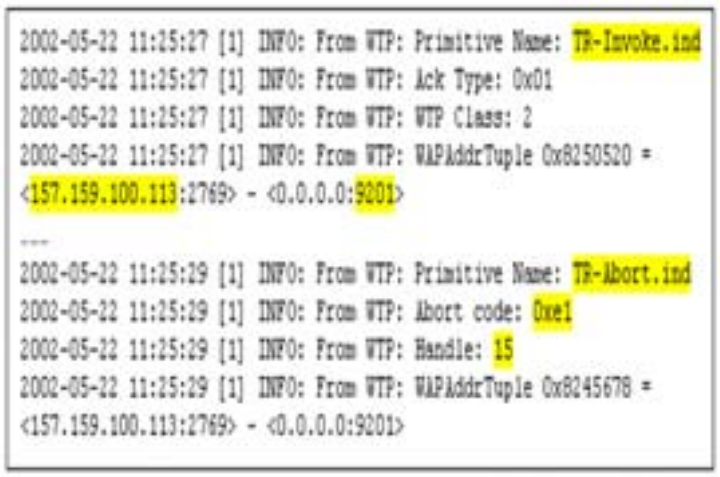


FIG. 5.11 – WSP PO avec le message injecté

(Validation par Contrôle et Observation des fichiers LOG) d'expérimentation en vue de valider l'interopérabilité entre une application sur un serveur et une application sur un terminal qui sollicite une passerelle pour communiquer avec le serveur.
Les implantations n'offrent pas d'interfaces directes. Nous avons donc défini des PCO et PO à travers une architecture, ce qui nous a permis de générer les fichiers suivants :

- un fichier nommé WSP_layer_PO.log qui représente une trace de toutes les interactions entre la couche *WSP_server* et les couches adjacentes,
- un fichier nommé WTP_layer_PO.log qui représente une trace de toutes les interactions entre la couche *WTP* et les couches adjacentes,
- un fichier nommé WesternWapper.log qui représente les traces des interactions entre la couche *WSP_client*, WTP et même WDP,
- et la visualisation grâce au sniffer ethereal des protocoles actifs dans le réseau.

A travers différents scénarios, nous avons constaté que l'implantation des couches WSP et WTP respectait bien la norme à travers les scénarios suivants :

- requête de classe 2 d'établissement d'une session avec phase de transaction dans le cas d'une requête avec une url correcte et dans le cas d'une requête avec une url erronée,
- transaction de classe 2 avec message abort du destinataire.

Notre architecture, à travers ses PCO et PO nous a permis de visualiser le transit global des évènements et d'en localiser les problèmes susceptibles de nuire à l'interopérabilité du système global comme nous l'avons démontré dans le cas d'une requête erronée.

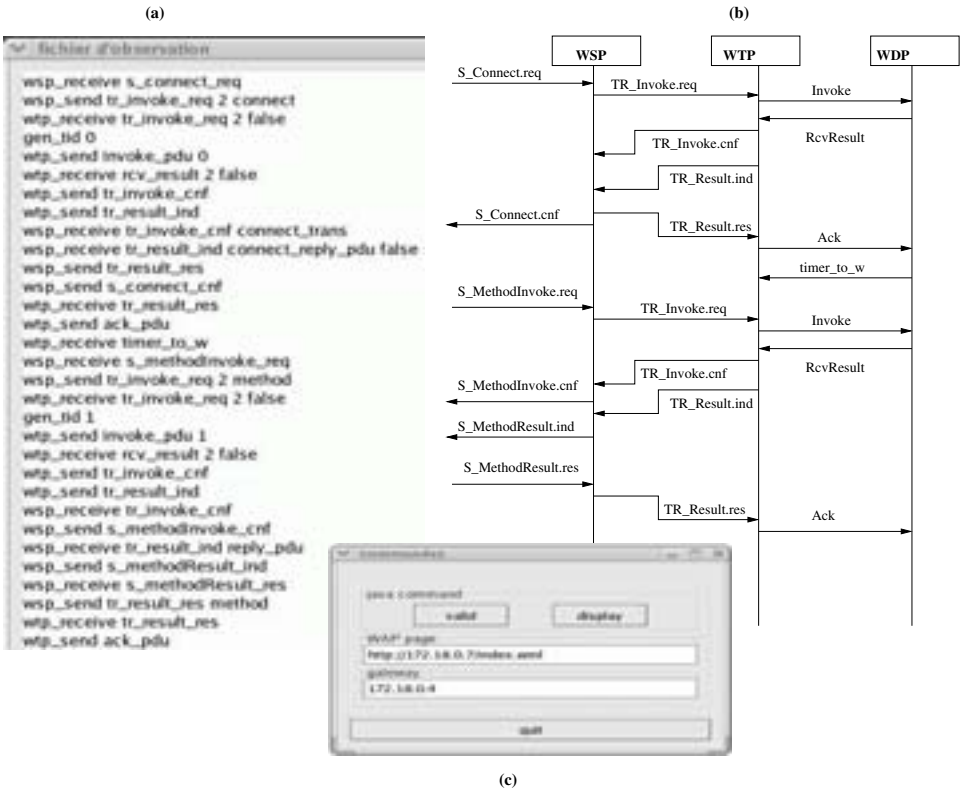


FIG. 5.12 – Fichier Log relatif à l'observation de WSP côté simulateur

Conclusion et perspectives

Contributions

La forte avancée observée dans le domaine des réseaux et plus particulièrement de l'Internet et des réseaux mobiles a inéluctablement contribuer à faire avancer l'idée de faire de l'Internet avec son mobile.
Faire de l'internet avec son terminal a nécessité la conception de nouveaux services et de nouveaux protocoles. Ces services et protocoles mettent en relation des entités hétérogènes, ce qui a posé entre autre le problème :

- de la garantie de leur interfonctionnement,
- de la nécessité de trouver des applications/services innovants capables de susciter l'intérêt des abonnés aux technologies mobiles.

Dans le cadre de ce travail de recherche, nous avons défini d'une part la validation comme le test des protocoles et des services et d'autre part nous avons adopté les normes WAP [28, 29] et UMTS [76] puis testé les services suivants :

- services applicatifs : ces services correspondent à l'exécution de programmes interagissant directement ou indirectement avec les utilisateurs finaux munis de leur mobile.
- services fournis par les couches protocolaires : ces services sont spécifiés dans les normes. Nous nous sommes intéressés aux couches protocolaires WSP et WTP.
- services innovant : nous avons défini comme services innovant, les services basés sur la localisation ou géo-dépendant. Ces services se positionnent comme une valeur de différentiation pour les opérateurs et mettent en oeuvre des mécanismes qui permettent au serveur de localisation de calculer les positions des mobiles afin de les fournir au serveur d'application sur demande autorisée.

Pour ces différents services et protocoles, nous avons défini une méthodologie de test basée sur les méthodes de génération et les architectures. La première étape de notre méthodologie a consisté en l'obtention d'une représentation précise des systèmes étudiés. Pour ce faire, nous avons utilisé le langage SDL [40] pour spécifier les protocoles et les services puis son corollaire MSC [39] pour spécifier les objectifs de test et représenter les scénarios de test. Les

tests se sont déroulés comme suit :

Le test d'application

Ces tests ont permis de vérifier qu'une application est bien écrite et répond aux attentes des utilisateurs finaux. Nous avons développé une méthodologie de test d'une application WML basée sur le reverse engineering et qui consiste à partir du code source vers un modèle SDL représentatif de l'application. Des tests sont ensuite générés à partir de ce modèle SDL en utilisant les méthodes classiques. Cette méthode présente l'intérêt de pouvoir s'appliquer à toute application écrite en WML contrairement à la méthode présentée dans [59].

Test de conformité des protocoles WSP et WTP puis des services de localisation

Il s'agit du test d'interopérabilité locale "verticale". Nous avons appliqué la stratégie de test développée pour vérifier et valider la conformité des protocoles WSP et WTP puis des services de localisation dans un réseau UMTS. Pour ce faire, nous avons dans un premier temps modélisé ces protocoles et services dans le formalisme SDL puis utilisé l'outil ObjectGeode dans les trois modes de simulation (aléatoire, interactive, exhaustive) afin de détecter les erreurs de conception et vérifier le comportement des modèles par rapport à certaines propriétés.
Les taux de couverture des transitions et des états ont été satisfaisants, nous n'avons pas décelé de situations de blocage, et avons vérifié sans problèmes les propriétés souhaitées. Cependant, nous avons relevé plusieurs ambiguïtés sur nos modèles (WSP, WTP et services de localisation) qui pourraient être à l'origine des problèmes futurs d'intéropérabilité. Ce qui nous a souvent conduit à interpréter la norme. Les scénarios engendrés, représentés sous le format MSC [39] seront utilisés sur une plate-forme réelle.

Concernant les services de localisation, plusieurs travaux [35, 64] ont été publiés sur l'analyse de performance, mais nous n'avons pas connaissance des études concernant la simulation fonctionnelle et la conception de tests basés sur des techniques formelles dans les réseaux UMTS notamment. Notre travail de recherche constitue donc une originalité en la matière. Nous pouvons ajouté à cela, les architectures de test proposées pour les différents protocoles et services qui ont fait l'objet de notre travail de recherche.

Test d'interopérabilité de l'architecture globale

Il s'agit de l'intéropérabilité distante "horizontale" des couches WSP et WTP que nous avons effectué grâce à la méthode que nous avons appelée VaICOLOG (validation de l'interopérabilité par Contrôle et Obser-

vation des fichiers LOG). C'est une méthode d'expérimentation en vue de valider l'intéropérabilité entre une application sur un serveur et une application sur un terminal qui sollicite une passerelle pour communiquer avec le serveur. Vu que les implantations que nous testons n'offrent pas d'interfaces directes, nous avons donc défini et implanté des PCOs et POs à travers une architecture. Ces POs nous ont permis de récupérer toutes les traces (WSP_layer_PO.log qui représente les traces de toutes les interactions entre la couche *WSP_server* et les couches adjacentes, WTP_layer_PO.log qui représente une trace de toutes les interactions entre la couche *WTP* et les couches adjacentes,WesternWapper.log qui représente les traces des interactions entre la couche *WSP_client*, WTP et même WDP, et la visualisation grâce au sniffer ethereal des protocoles actifs dans le réseau) relatives à l'exécution d'une requête sur un terminal et la réception de celle-ci sur le serveur en passant par la passerelle. Notre architecture, à travers ses PCOs et POs nous a permis de visualiser le transit global des évènements et d'en localiser les problèmes susceptibles de nuire à l'interopérabilité du système global.

L'originalité de notre plate-forme réside dans le fait qu'il s'agit d'une plate-forme de validation et d'expérimentation de l'interfonctionnement des protocoles et services. A notre connaissance, les plate-formes existantes en France (VTHD : plate-forme d'interconnexion des réseaux à très haut débit, Amarrage : plate-forme multimédia basée sur des réseaux actifs, Parol : plate-forme basée sur l'ORB Jonathan, @IRS : Architecture Intégrée de Réseaux et Services) ou dans le monde (à savoir les plate-formes de l'ETSI, de Corée Telecom et de l'Université de Tsinghua en Chine) sont davantage destinées à permettre des développements ou des mises en oeuvre de services et non pas à la validation d'interfonctionnement globale de protocoles et de services.

Perspectives

Les réseaux had hoc

Nous envisageons d'étendre l'application de nos techniques à d'autres types de réseaux mobiles, comme par exemple, les réseaux ad hoc. Ces réseaux ne requièrent pas d'administration centralisée ou une infrastructure de réseau fixe comme par exemple les stations de base ou les points d'accès. De plus, une configuration dynamique du réseau est nécessaire pour réagir aux changements éventuels du contexte. Toutes ces caractéristiques rendent indispensable l'adaptation et la mise en place de nouvelles méthodes de test.

Le modèle de performance

On constate que 80% des systèmes client/serveur nécessitent d'être reconçus du fait que les exigences de performance ne sont pas satisfaites, alors qu'une étude de performance coûte généralement moins de 3% du budget complet du projet de développement. Les ressources étant de capacité limitée, les tâches vont entrer en concurrence pour l'accès aux ressources, ce qui est résolu par des mécanismes d'ordonnancement que l'on associe à des files d'attente. Afin d'intégrer la vérification fonctionnelle et l'évaluation des performances, il est possible d'envisager une démarche permettant la dérivation d'un modèle de performance à partir d'un modèle fonctionnel.

Annexe A

Application MobInfo

Cette application est sensée mettre en application l'acquisition de l'information du mobile afin de rendre un contenu personnalisé. Pour le moment, nous renseignons nous mêmes la position du mobile. Ci dessous, quelques écrans qui présentent le rendu de l'application.

A.1 Accueil et menu principal

L'utilisateur accède à l'application via son terminal, il choisit *Local Info* dans la liste des services proposés par son opérateur. Le logo du service apparaît puis au bout de quelques secondes, l'écran du menu s'affiche (figure A.1, en haut à gauche), proposant un accès direct aux différentes fonctions de l'application. On a alors le choix entre :

- Consulter l'itinéraire,
- Consulter les points de proximité,
- Consulter le trafic urbain,
- Demander la position courante,
- Effectuer une recherche.

A.2 Itinéraire

Les itinéraires aident les abonnés à planifier leurs voyages, à gérer leur temps et à atteindre leurs destinations en décrivant au mieux l'itinéraire et le temps de voyage selon l'état du trafic courant.
En sélectionnant l'itinéraire (figure A.1), une boîte de dialogue nous permet de renseigner notre destination. Après validation de la destination saisie, il nous est demandé de préciser notre moyen de transport pour ensuite recevoir un détail du trajet que nous devions parcourir rue par rue ou métro par métro selon le moyen de transport choisit.

FIG. A.1 – Itinéraire

A.3 Proximité

Les services de proximité permettent aux abonnés de rechercher et/ou de repérer dans leur voisinage les points tels que : *station, hôpitaux, parkings, arrêts de bus, musées, monuments, services publics, etc.*
Le choix du sous-menu proximité (figure A.2) permet à l'utilisateur de visualiser la liste des points précédents. En sélectionnant un point de proximité (par exemple le cinéma puis éventuellement le type de cinéma), on obtient l'adresse précise et la distance à parcourir. Il est possible par suite de demander l'itinéraire.

A.4 Trafic

Les cartes de trafic aident les abonnés à éviter l'encombrement en affichant l'information à jour du trafic. La sélection de ce sous-menu (figure A.3) nous donne la possibilité d'accéder au trafic en visualisant une carte dédiée. Un zoom peut être effectué afin de mieux visualiser des zones particulières.

FIG. A.2 – Proximité

A.5 Recherche et position courante

Le module de recherche A.4 permet à l'utilisateur d'effectuer une recherche par mot-clé (par exemple un nom de rue, une place, etc..). La chaîne de caractère saisie est recherchée dans tous les points (rue, restaurant, etc..) contenus dans la base de données. On obtient le résultat sous la forme de carte de trafic ou d'itinéraire, selon le besoin. Ce module nous donne aussi la possibilité de demander notre position si on est dans un endroit qui nous est étranger.

FIG. A.3 – Trafic

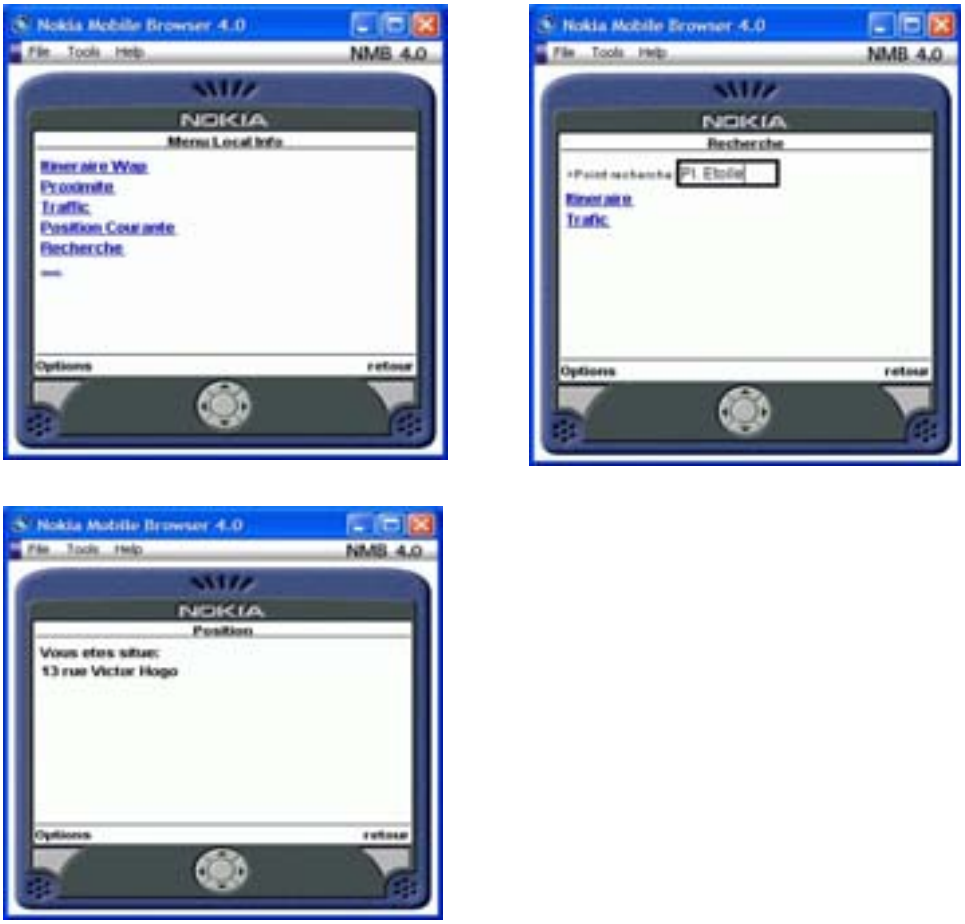


FIG. A.4 – Détermination de la position et recherche d'un point

Annexe B

Mise en place de la plate-forme

B.1 Les logiciels expérimentés côté client

Les sous-sections qui suivent présentent les différents outils utilisés pour le développement d'une application en WML et ceux utilisés pour la mise en place de la plate-forme.

B.1.1 Les éditeurs et générateurs de code

En théorie, un éditeur aussi rudimentaire que NotePad est tout à fait suffisant pour écrire du WML, mais en pratique, cela devient rapidement fastidieux, et il apparaît alors indispensable de pouvoir disposer d'un logiciel qui vous propose des modèles de decks vierges, contenant déjà les éléments obligatoires, -comme la déclaration de la version de XML -, et qui vous permet d'insérer les balises automatiquement. Nous avons expérimenté l'éditeur suivant :

- *Waptor de WAPDrive* : c'est ce que propose Waptor, dans une interface assez ergonomique, avec une mise en couleur du code, qui permet de distinguer tout de suite : balises, attributs, texte et variables, et affichage approximatif du résultat dans une fenêtre adjacente. Il se révèle très pratique à l'usage, notamment grâce à la présence d'icônes d'insertion de balises pour les fonctions les plus courantes.

B.1.2 Les navigateurs

Les navigateurs nous permettent de consulter d'une part les sites Wap existants et d'autre part, ils nous permettent de tester nos propres fichiers WML pour contrôler la correction et le rendu final. Ci-dessous les navigateurs que nous avons expérimentés :

- *Wireless Companion* : il comprend à la fois un navigateur WEB classique et un navigateur WAP qui a l'avantage d'émuler les différents téléphones mobiles du marché.

FIG. B.1 – M3Gate

- *M3Gate* : ce navigateur propose deux formats d'affichage, l'un sous forme mobile, l'autre de PDA. Il est très pratique et simple d'utilisation.

B.1.3 Les Kits de développement

La plupart des kits de développement comprennent entre autres un éditeur et un navigateur. Ils sont très réalistes car ils reproduisent non seulement l'apparence mais aussi le fonctionnement du téléphone existant. A cet effet, nous avons expérimenté les kits suivants :

- *Ericsson WapIDE* : l'espace de travail de cet outil de développement se compose de trois fenêtres, une pour l'éditeur, une pour le simulateur et une qui affiche la sortie du compilateur.
- *Nokia Wap Toolkit* : ce kit se compose de deux fenêtres, l'une pour gérer votre code WML, et l'autre pour afficher le résultat. La fenêtre d'édition est composée de cinq onglets. Nous disposons d'un onglet dans lequel se trouve le code édité et un autre pour connaître la sortie du compilateur. Ce Kit dispose également d'un programme graphique permettant de créer des images au format WBMP.

FIG. B.2 – Nokia WAP Toolkit

- *UP.SDK4.0 de phone.com* : il permet de simuler non seulement les capacités techniques d'un téléphones, comme la disposition des touches ou la taille de l'écran, mais également le comportement interne du téléphone.

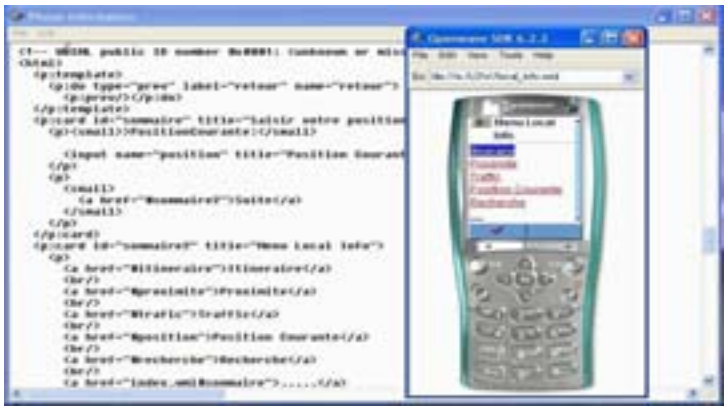


FIG. B.3 – UP.SDK

B.2 Le serveur

La présente section précise les différentes applications et outils utilisés lors de la mise en place des serveurs de données. Nous avons expérimenté un serveur sous le système linux et un serveur sous Windows avec les éléments décrits ci-dessous :

B.2.1 Configuration sous Linux

- *le serveur Web* : sous linux, nous avons utilisé le serveur apache. Il est disponible sur le site : www.apache.org.
- *outil de développement des scripts* : comme outil de développement des scripts côté serveur nous avons opté pour l'usage de PHP. PHP est disponible sur le site : www.php.net.
- *la base de données* : la base de données utilisée est MySQL, disponible sur le site : www.mysql.com.

B.2.2 Configuration sous Windows

- *le serveur Web* : nous avons utilisé Internet Information Server 4.0 de Microsoft. Son principal intérêt est de disposer en standard d'un moteur de script permettant de générer des pages dynamiques : Active Server Page (ASP). La technologie ASP repose sur l'exécution des scripts côté serveur.
- *outil de développement des scripts* : ASP
- *la base de données* : la base de données utilisée est *Microsoft Access* en raison de sa simplicité de mise en oeuvre.

B.2.3 Serveurs et types MIME

La publication de pages WML s'effectue comme pour les pages WEB ; elles sont téléchargées sur un serveur, souvent avec le protocole FTP. Les fichiers sont généralement suivis de l'extension **.wml*. Le tableau ci-dessous vous permet de déterminer, à partir d'une extension fichier, le type de document dont il s'agit ainsi que son type MIME, nécessaire pour configurer correctement le serveur Web.

Contenu	Type MIME	Extension fichier
Fichier WML	text/vnd.Wap.wml	Wml
WMLScript	text/vnd.Wap.wmlscript	Wmls
WML compile	application/vnd.Wap.wmlc	Wmlc
WMLScript compile	application/vnd.Wap.wmlscriptc	Wmlsc
Image WBMP	image/vnd.Wap.wbmp	wbmp

B.3 Installer un serveur PPP sur PC Linux

C'est une opération importante dans la mise en oeuvre de notre plateforme, nous en parlons car cela n'a pas été facile et relève plus d'une procédure de tâtonnement. Nous n'avons pas trouvé de documentation clé en main. Notre expérience peut donc servir et constituer un document en la matière.

B.3.1 Paquetages nécessaires

Sur le PC qui va servir de serveur PPP, il est nécessaire d'avoir un modem (dans notre cas, nous avons utilisé *Speed'Com V92* de OLITEC) connecté et d'avoir installé les paquetages suivants :

- *ppp* : il est bien entendu nécessaire d'installer un paquetage *ppp*, notons qu'il est pré-installer dans les versions actuelles de linux.
- *mgetty+sendfax* : ce paquetage permet de gérer les connexions (login) sur une ligne série via un modem. On se sert donc de *mgetty* pour répondre à la connexion téléphonique établie par le client PPP. Dans ces grandes lignes, *mgetty* va :
  - décrocher la ligne et répondre à l'appel,
  - envoyer certaines demandes d'informations, puis,
  - lancer *pppd* pour établir la connexion réseau IP.
- *ipfwadm (ou ipchains pour les noyaux Linux* $>$ 2.2.0*)* : pour gérer le routage des paquets IP depuis le serveur PPP vers le reste de l'Internet. Sans cela le PC serveur PPP n'offrira sans doute pas la connectivité totale au reste de l'Internet.

B.3.2 Gérer les appels entrants sur le serveur PPP, avec mgetty

Après avoir installé le paquettage *mgetty+sendfax*, les fichiers de configurations de *mgetty* sont dans */etc/mgetty+sendfax* :

- *le fichier dialin.config* contrôle l'acceptation ou le refus des appels téléphoniques entrants, selon le numéro de téléphone de l'appelant.
- *le fichier login.config* contrôle comment et quand mgetty doit lancer un autre programme lors de la connexion d'un utilisateur, et ceci en lieu et place du login normal.
- *le fichier mgetty.config* contrôle le paramétrage de la ligne série *ttyS0* sur lequel est branché le modem du serveur. Il est juste nécessaire de modifier les lignes de paramètres correspondant au port série *ttyS0* sur lequel est branché le modem.

B.3.3 Lancer et Automatiser le lancement de mgetty

On va se servir du process init de linux et de son fichier de paramètres */etc/inittab* pour automatiser le lancement de mgetty. Pour cela il suffit de rajouter une ligne dans le fichier */etc/inittab* qui va permettre de relancer automatiquement le process mgetty par init.

mgetty est lancé avec *l'option -D* sur la ligne série *ttyS0*, ce qui signifie que la ligne série et le modem doivent être traités comme un modem échangeant des données (et pas en mode fax) par : *g1 :345 :respawn :/sbin/mgetty -D /dev/ttyS0* .

Sep 6 00 :11 :35 compc32 mgetty[4628] : cannot get terminal line dev=ttyS1, exiting : Erreur d'entrie/sortie.
(l'erreur dans ce cas est d'avoir mis ttyS1 à la place de ttyS0)

08/04 09 :25 :43 yS0 timeout in chat script, waiting for 'RING'
08/04 09 :25 :43 yS0 huh ? Junk on the line ?
–
08/04 09 :26 :17 yS0 timeout in chat script, waiting for 'RING'
08/04 09 :26 :17 yS0 huh ? Junk on the line ?
–
$08/0409 : 29 : 47\sharp\sharp\sharp\sharp\sharp failed dev = ttyS0, pid = 2577, got signal 15, exiting$

exemple d'une bonne connexion à 22 :56, L'utilisateur "libesppp" s'est bien connecté
$08/0422 : 56 : 04\sharp\sharp\sharp\sharp\sharp data dev = ttyS0, pid = 1772, caller =' none', conn =' 9600/ARQ/V34', name ='', cmd =' /bin/login', user =' libesppp'$

FIG. B.4 – Exemple de messages obtenus

En rajoutant cela dans le fichier */etc/inittab* et en demandant la relecture de ce fichier par le démon *init* (par */sbin/init q*), ça va lancer le processus mgetty en mémoire et initialiser le port ttyS0 pour les besoins du modem.

Après avoir lancé le processus mgetty en mémoire, il faut controler tout de suite si cette initialisation s'est bien passée. Pour cela, il convient de regarder le fichier de trace relatif à *mgetty* qui se trouve dans */var/log/mgetty.log/ttyS0*, ainsi qu'à */var/log/messages*. S'il ya des erreurs, inutile d'aller plus loin, cela signifie que le modem s'initialise mal et il est alors nécessaire d'en trouver la raison. Un exemple de messages obtenus est présenté dans la figure B.4 :

A ce niveau, s'il n'y a pas eu d'erreur, *mgetty* est lancé par *init* et est en attente d'appels téléphoniques.
Si le client *ppp* appelle, *mgetty* va répondre (modem décroche) ; *mgetty* va envoyer des chaines pour demander une authentification. Si l'authentification est correcte (vérification dans */etc/passwd*), le "user" X sera alors connecté. En se connectant */usr/sbin/pppd* est lancé en tant que shell de connexion, le démon pppd est alors lancé (ligne */AutoPPP/ - a_ppp /usr/sbin/pppd* auth $-chap + pap$ login modem debug crtscts $ttyS038400$ présente dans le fichier login.config), et l'authentification "pap" a alors lieu (mots de passe de pap-secret), la liaison réseau IP entre le client et le serveur sur RTC est

alors établie.

B.3.4 L'authentification des "connexions" sur la liaison PPP

Pour qu'un utilisateur X puisse se connecter, il est nécessaire de créer des comptes d'accès au réseau pour les utilisateurs se connectant depuis le PC client. Pour cela il faut créer de nouvelles entrées dans le fichier */etc/passwd* du coté serveur.
Ainsi pour un accès pour l'utilisateur *"libesppp"* on créera un compte spécial dans *"/etc/passwd"*. Le shell du login est remplacé par le lancement de pppd :
libesppp :LO2QP457L8RXc :801 :801 :compte PPP pour libes :/home/ppp :/-usr/sbin/pppd

Bien sûr, le programme pppd devra pouvoir être lancé par n'importe quel utilisateur. Il faut donc vérifier les permissions pour exécuter pppd. Le mieux est de placer le bit suid root par *chmod u+s /usr/sbin/pppd*.

mgetty envoie par défaut au client (cela dépend de comment il a été compilé) le nom IP du PC serveur, suivi de "login :", chaine à laquelle il faudra répondre par un nom de connexion Unix valide (c'est à dire un qui figure dans /etc/passwd). Puis dans le deuxième échange mgetty envoie la chaine "passwd" à laquelle il faut répondre par le password de l'utilisateur qui se connecte par le RTC.

Annexe C

Modélisation SDL de WTP destinataire

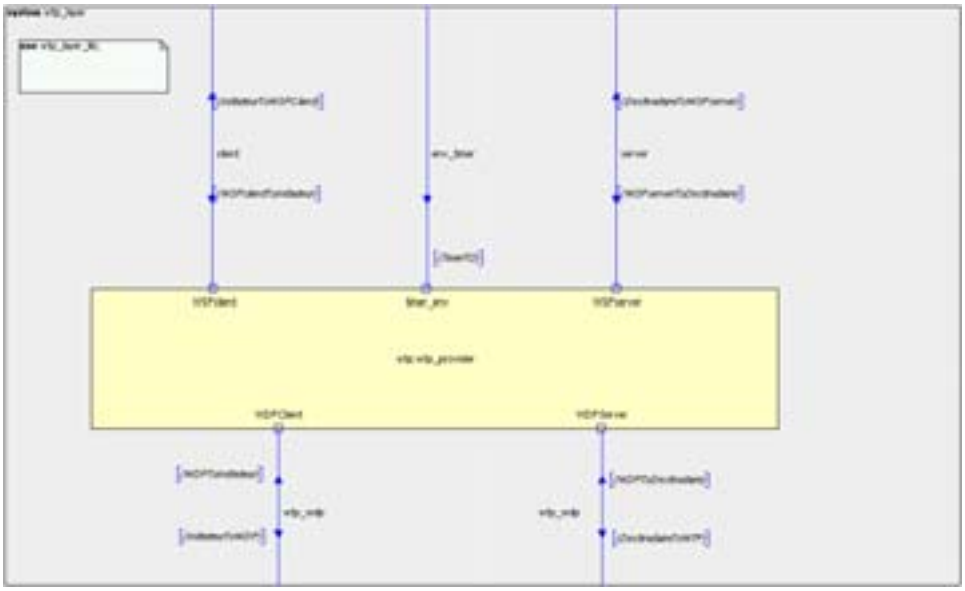


Fig. C.1 – Système WTP

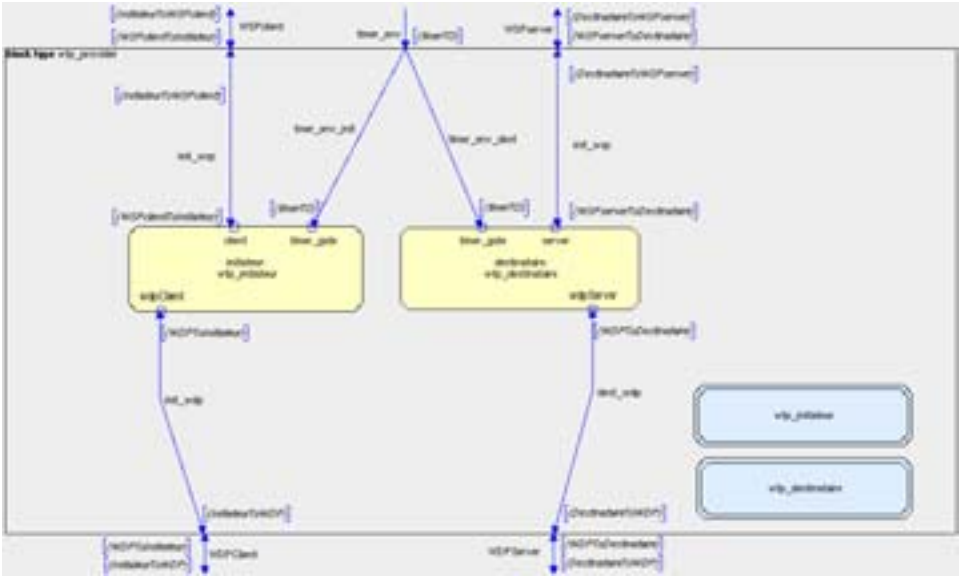


FIG. C.2 – WTP Bloc Provider

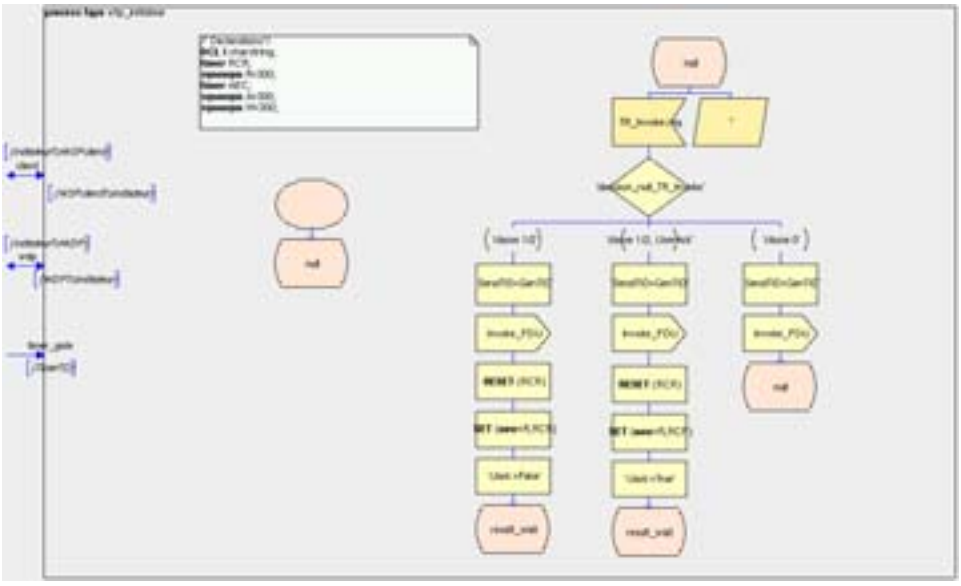


FIG. C.3 – WTP process initiateur : état null

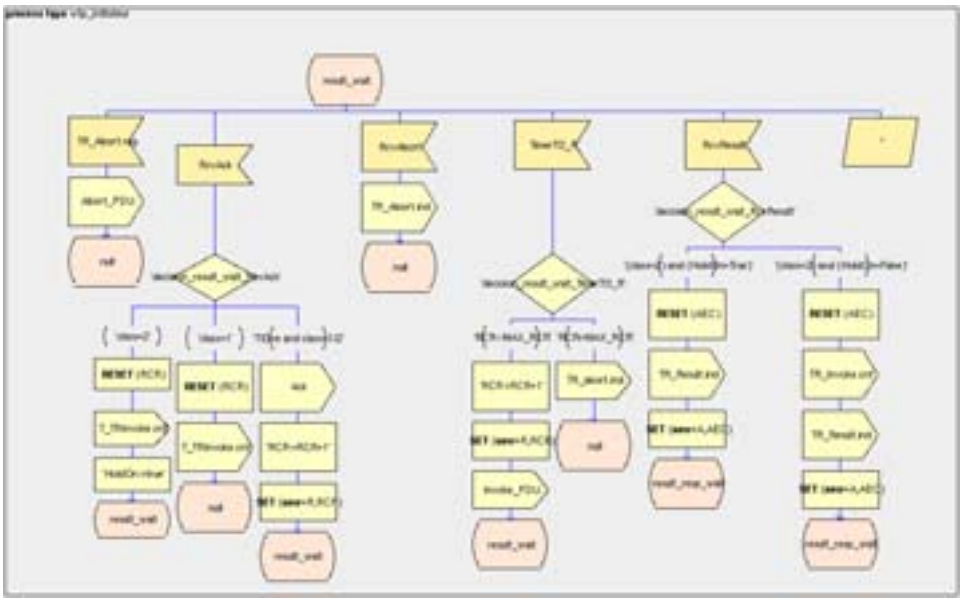


FIG. C.4 – WTP process initiateur : état result_wait

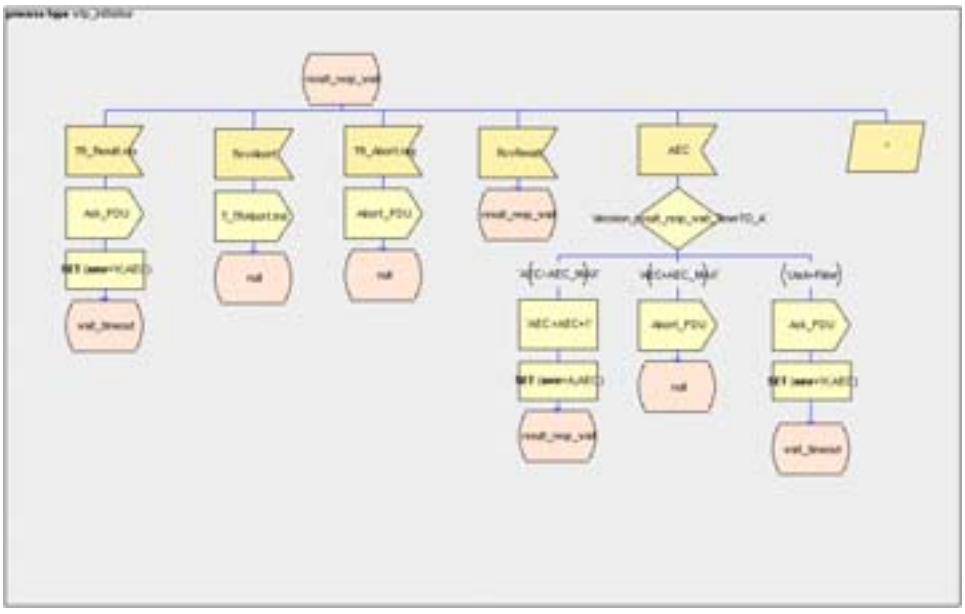


FIG. C.5 – WTP process initiateur : état result_resp_wait

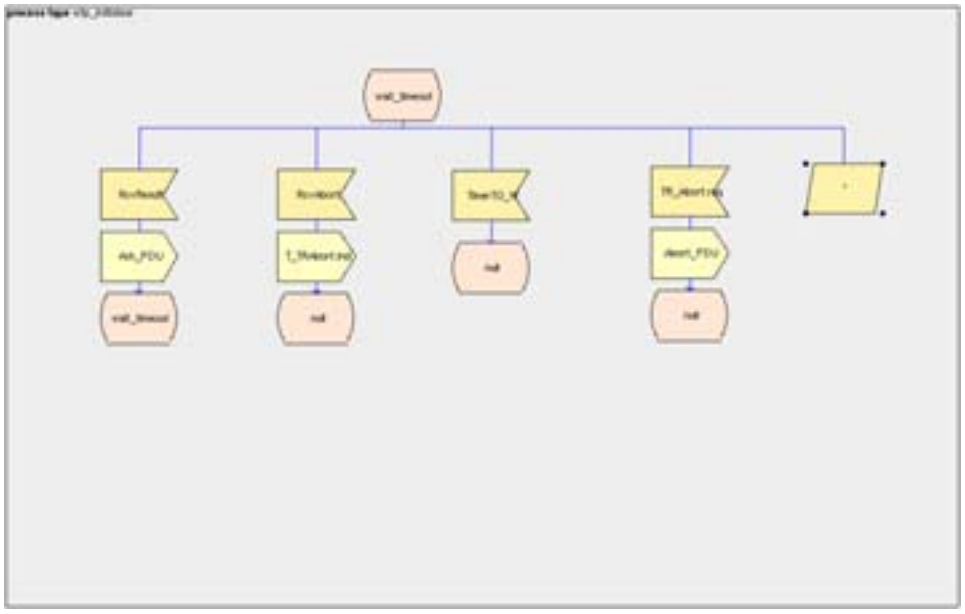


FIG. C.6 – WTP process initiateur : état wait_timeout

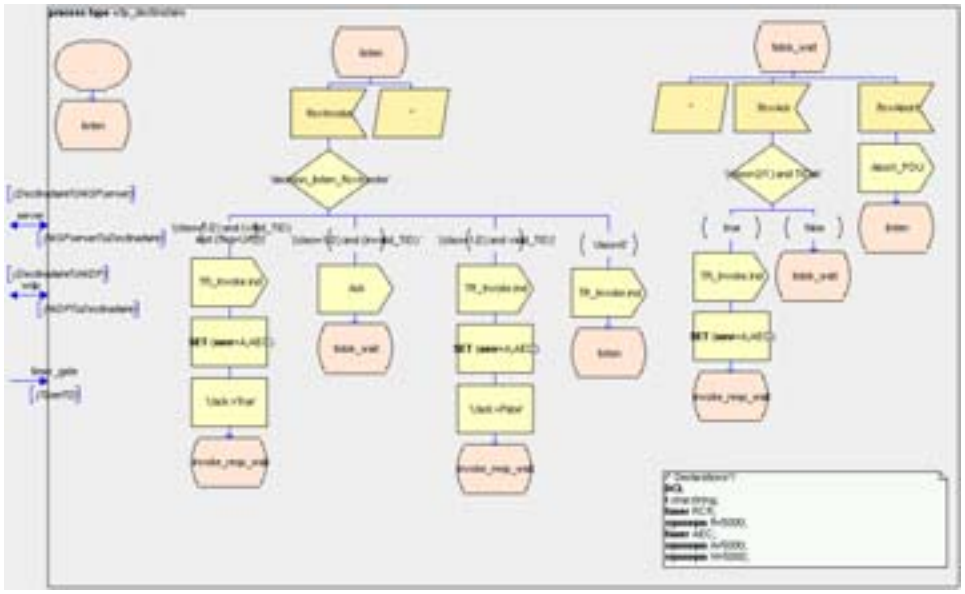


FIG. C.7 – WTP process destinataire : états listen et tidok_wait

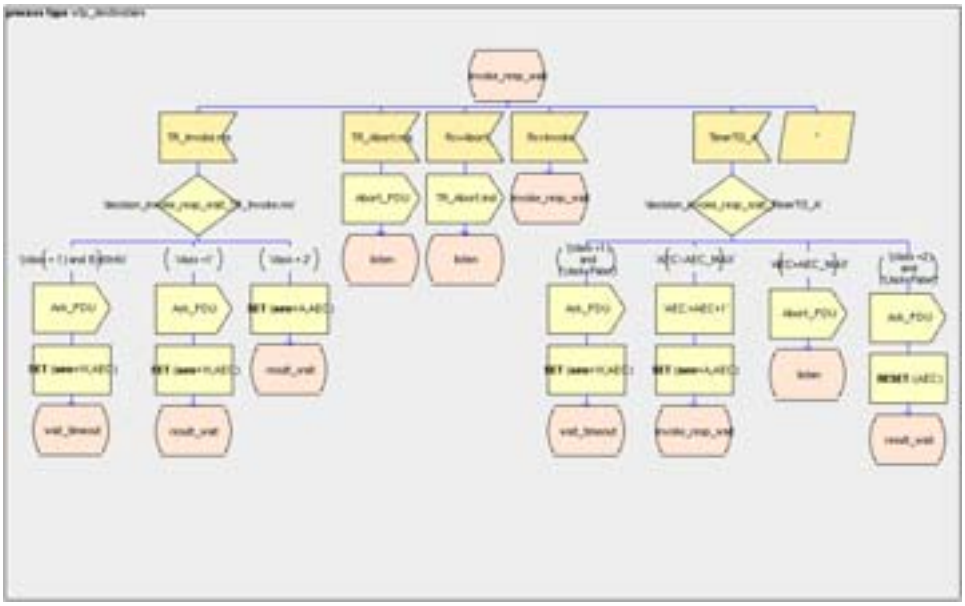


FIG. C.8 – WTP process destinataire : état invoke_resp_wait

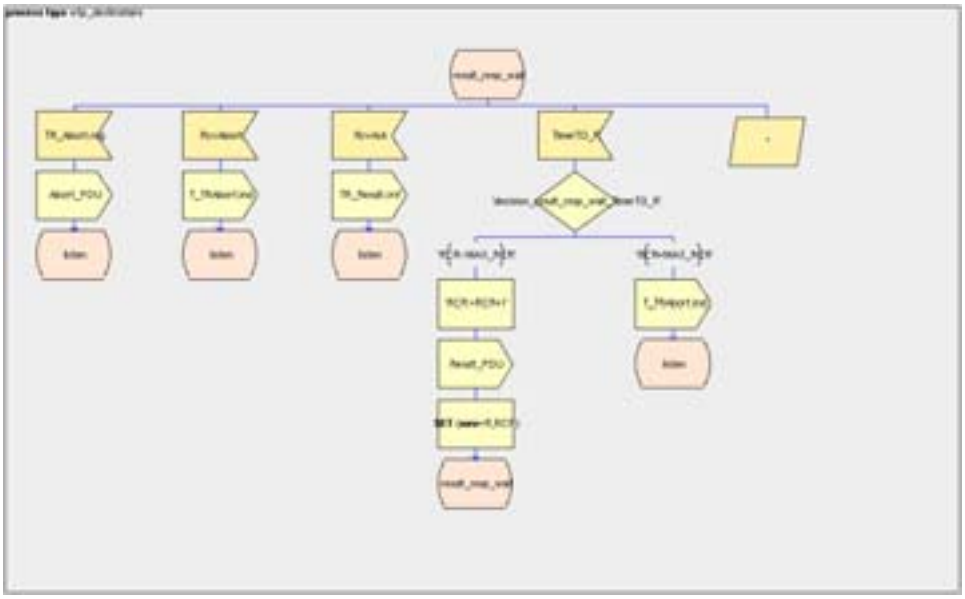


FIG. C.9 – WTP process destinataire : état result_resp_wait

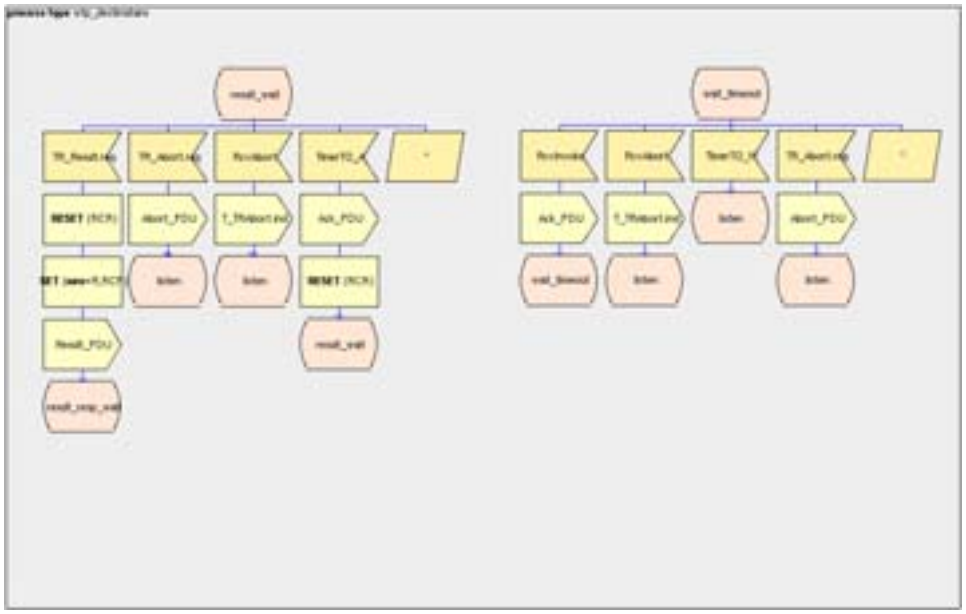


FIG. C.10 – WTP process destinataire : états result_wait et wait_timeout

Annexe D

Programmation des points d'observation sur la passerelle

Une interface graphique a été implémentée afin de pouvoir afficher les observations faites à travers Kannel. Cette interface permet d'une part la connexion des composants de la passerelle, à savoir la Bearer box et la Wap box, et d'autre part l'ouverture d'une fenêtre permettant d'observer le contenu des fichiers. Par conséquant, trois boutons sont dédiés aux fonctionnalités précédentes : "beare box", "wapbox" et "valid". Le choix des fichiers LOG à afficher se fait optionnellement à l'aide de boutons radio.
Afin de respecter l'architecture de la passerelle, l'interface a été dévellopée en langage C en utilisant les librairies GNU Toolkit (GTK). Les codes intégrés à la passerelle contiennent les fonctions suivantes :

- fonctions de création des objets graphiques :
  - *create_execution()* crée les boutons de l'interface,
  - *create_couche()* crée les boutons radio,
  - *insert_text()* transforme le contenu des fichiers en objet texte lu par GTK,
  - *create_text()* crée l'objet qui permet d'afficher le texte, et donc appelle *insert_text()*,
  - *main()* affiche la fenêtre de commandes, appelle *create_couche()* et *create_execution()*. Cette fonction crée également le bouton "quit" qui permet de quitter l'application.
- fonction de gestion des évènements :
  - *callback_click_WSP* et *callback_click_WTP* sont des fonctions appelées par les boutons radio en vue du choix des fichiers LOG à afficher,
  - *callback_bearerbox()* et *callback_wapbox()* permettent de lancer les composants de la passerelle,

- *callback()* crée une nouvelle fenêtre pour afficher le contenu du fichier LOG. Elle fait appel à la fonction *create_text()*,
- *delete_event()* ferme la fenêtre de visualisation,
- *close_application()*, ferme l'application par le bouton "quit".

FIG. D.1 – Interface graphique de kannel

Publications

Notre participation au projet PLATONIS dans le cadre de ce travail a donné lieu aux publications suivantes :

Revues internationales

R. Castanet, S. Chaumette, M. Mackaya and the Platonis consortium. *Applications and services in wireless networks*. Hermes Penton Science. Pages 217-229, 2002.

Revues nationales

R. Castanet, M. Mackaya, P. Combes, Wei Monin, A. Cavalli, A. Mederreg, F. Zaidi, P. Laurençot. *Une plate-forme de validtion multi-protocoles et multi-services-Résultats d'expérimentation*. Annales des Télécommunications. 2005.

Colloques internationaux avec comité de sélection et actes

A. Cavalli, R. Castanet, S. Chaumette, M. Mackaya, and al., *The Platonis Project*, IEEE First International Workshop on Services Applications in the Wireless Public Infrastructures, Mai 2001, Evry, France.

M. Mackaya, O. Koné, R. Castanet, *Modelling Locations Operations in UMTS Networks*, 8th ACM International Conference on Mobile Computing and Networking, September 23-28, 2002, Atlanta, Georgia, USA

M. Mackaya, A. Meddereg, *Test des systèmes mobiles*, GRES'03 Colloque Francophone sur la Gestion de Réseaux Et de Services 24-27 février 2003, Fortaleza, Brésil.

M. Mackaya, R. Castanet, *Modelling and Testing Location Based Application in UMTS Networks*, IEEE Contel 2003, 7th International Conference

on Telecommunications, June 11-13, 2003, Zagreb,Croatia

A. Cavalli, R. Castanet, M. Mackaya and al., *Une plate-forme de validtion multi-protocoles et multi-services-Résultats d'expérimentation*, CFIP 2003, Colloque Francophone sur l'Ingenierie des Protocoles. Paris, 7-10 Octobre 2003, France

A. Cavalli, R. Castanet, M. Mackaya and al., *A Multi-Protocol Validation Platform - Experimentation Results*, TestCom 2004, The 16th IFIP International Conference on Testing of Communicating Systems, 17-19 March 2004, St Anne's College, Oxford, England

Bibliographie

[1] ISO 9646-3. Method for testing and specification (MTS) ; the testing and test control notation, version 3 ; part 2. In *TTCN-3 core Language*. ETSI, 2001.

[2] J.M. Autebert. Langages algebriques. In *Collection Etudes et Recherche en Infomatique*, 1987.

[3] J.P. Baconnet, C. Betteridge, G. Bonnes, F. Van den Berghe, and T. Hopkinson. Scoping further ewos activity for interoperability testing. In *Technical Report EGCT/96/130 R1*. EWOS, September 1996.

[4] S. Barbin, L. Tanguy, and C. Viho. A step towards a formal framework for interoperability testing. In *Publication interne N° 1408*. IRISA, Juillet 2001.

[5] J.L. Batard, J.D. Colas, P. Corvisier, E. Mansord, and A. Woog. Trace analysis for conformance and arbitration testing. In *IEEE Transaction on software engeneering*, volume 15, pages 1347–1356, November 1989.

[6] Y. Benkhellat. Formalisation et verification de l'interoperabilite dans les systemes de communication. In *PhD thesis, INPL Institut Polytechnique de Loraine*, 1995.

[7] G. Bochmann, R. Dssouli, and B. Sarikaya. Architecture et sélection de test. In *CFIP'88*, 1988.

[8] G.V. Bochmann and al. Fault model in testing. In *Proc. 4th International Workshop on Protocol Testing Systems. Leidschendam, The Netherlands*, 1991.

[9] E. Brinksma. A theory fr the derivation of tests. In *Proc. 8th IFIP symposium on Protocol Specification, Testing and Verification*, 1988.

[10] Y. Byun, B.A. Sanders, and Chang-Sup Keum. Design patterns of communicating extended finite state machine in sdl. In *PLoP 2001 Conference*, 2001.

[11] R. Castanet, P. Corvisier, and C. Casadessus. Industrial experience on test suite coverage measurement. In *5th International Workshop on Protocol Testing Systems*, 1992.

[12] R. Castanet and O. Koné. Deriving coordinated testers for interoperability. In *Protocol Test System, VI (C-19), Elsevier Science Publ B. V. (North-Holland)*, pages 331–345, 1994.

[13] R. Castanet and O. Koné. The tbroker platform for the interoperability of communication systems. 2001.

[14] A. Cavalli, Castanet, M. Mackaya, and al. *PLATONIS : a platform for validation and experimentation of multi-protocols and multi-services*. Hermes Penton Science, 2002.

[15] A. Cavalli, R. Castanet, M. Mackaya, and al. Une plate-forme de validation multi-protocoles et multi-services - résultats d'expérimentation. In *Proc. of CFIP 03*, Octobre 2003.

[16] A. Cavalli, R. Castanet, M. Mackaya, and al. A multi-protocol validation platform - experimentation results. In *TestCom 2004, The 16th IFIP International Conference on Testing of Communicating Systems*, 17-19 March 2004.

[17] O. Charles. Application des hypothèses de test à une définition de la couverture. In *Thèse de doctorat de l'Université Henry Poincarré*, Octobre 1997.

[18] O. Charles and R. Groz. Formalisation d'hypotèses pour l'évaluation de la couverture de tests. In *CFIP'96*, 1996.

[19] T. Chow. Testing software design modelled by finite state machine. In *IEEE Transaction on Software Engineering*, May 1978.

[20] M. Clatin, R. Groz, M. Phalippou, and R. Thummel. Two approaches linking a test generation tool with verification techniques. In *Proc. IWPTS, Evry, France*, 1995.

[21] E. Conquet and M. Perrotin. Sdl contest, railway crossing specification. In *SAM Workshop*, 2002.

[22] L. Doldi, V. Encontre, J.C. Fernandez, T. Jeron, S. Le Bricquir, N. Texier, and M. Phalippou. Assesment of automatic generation of conformance test suites in an industrial context. In *Proc. IWPTS, Evry*, 1995.

[23] K. Drika. Transformation et composition de graphes de refus : analyse de la testabilité. In *Thèse de doctorat, Université Paul Sabatier*, 1992.

[24] R. Dssouli and J. Zhao. Réunion interopérabilité, compte rendu. In *Comité Technique de l'ACERLI*, Février 1991.

[25] O. Dubuisson. Asn.1. In *Springer*, 1999.

[26] M. Diaz Edited by P.H. Van Eijk, C.A. Vissers. The formal description technique LOTOS. In *Esprit/ Sedos project*, 1989.

[27] J.C. Fernandez, C. Jard, and C. Viho. Using-on-the-fly verification techniques for generation of test suites. In *CAV'96. LNCS 1102, Springer Verlag*, 1996.

[28] WAP Forum. Eyrolles, 2001.

[29] WAP Forum. Wap specification. In *http ://www.wapforum.org*, 2004.

[30] S. Fujiwara and al. Test selection based on finite state models. In *IEEE Trans. on Software Engineering*, pages 591–603, 1991.

[31] J. Gadre, C. Roher, C. Summers, and S. Symington. A cos study of osi interoperability. In *Computer Standards and Interfaces*, volume 9, pages 217–237, 1990.

[32] A. Gill. Introduction to the theory of finite state machines. In *Mc graw hill, New York*, 1962.

[33] J. Grabowski. Test case generation and test case specification with mesage sequence chart. In *PhD thesis, University of Berne*, 1994.

[34] N. Griffeth, R. Hao, D. Lee, and R.K. Sinha. Integrated system interoperability testing with applications to voip. In *CFIP'96*, pages 14–17, October 1996.

[35] H. Holma. A study of umts terrestrial radio access performance. In *PhD Thesis*, 2003.

[36] D. Hopcroft and J. Ullman. Introduction to automata theory, languages and computation. In *Addison-Weley publishing company*, 1979.

[37] M. Ionescu. Une stratégie de test en télécommunications. In *Thèse de doctorat de l'Université Evry Val d'Essone*, 2001.

[38] ISO/IEC. Information technology. In *Open Systems Interconnection Conformance Testing methodology and framework*, 1991.

[39] ITU-T. Message sequence chart (MSC), recommandation Z.120. November 1999.

[40] ITU-T. Specification and description language. In *Recommandation Z.100, http ://www.sdl-forum.org*, 1999.

[41] C. Jard and T. Jéron. Bounded memory algorithms for verification on-tht-fly. In *CAV'91 : Symposium on Computer Aided Verification*, volume 575 of LNCS, pages 192–202. Springer-Verlag, 1989.

[42] C. Jard and T. Jéron. On-line model-checking for finite linear temporal logic specifications. In *International Workshop on Automatic of LNCS*, pages 275–289. Springer-Verlag, 1989.

[43] C. Jard, T. Jéron, J.-C. fernandez, and L. Mounier. On the fly verification of finite transition systems. In *Rapport de recherche N° 1861*, février 1993.

[44] T. Jéron. *TGV : Théorie, principes, et algorithmes.* 2002.

[45] S. Kang and M. Kim. Test sequence generation for adaptative interoperability testing. In *Proc. Protocol Test System*, volume 3, pages 187–200, 1995.

[46] A. Kerbrat, T. Jéron, and R. Groz. Automated test generation from sdl specifications. In *SDL'99*. Elsevier, 21-25 June 1999.

[47] Z. Kohavi. Switching and finite automata theory. In *MC GRAW-HILL*, 1978.

[48] O. Koné and R. Castanet. *Méthodes formelles de test de conformité des protocoles*, volume 18. Hermes Science, 1999.

[49] G. Leduc. Equivalence associée à la relation de conformité et simplification du testeur canonique en lotos. In *CFIP91*, pages 425–440. Hermes, 1978.

[50] G. Leduc. Testing software design modelled by finite state machine. In *Dissertation d'agrégation de l'Université de Liège*, 1990.

[51] G. Leduc. A framework based on implementation relations for implementing LOTOS specifications. In *Computer Networks and ISDN Systems 25*, pages 23–41, 1992.

[52] D. Lee and M. Yannakakis. Testing finite-state machines : State identification and verification. In *IEEE Transaction on Computers*, pages 306–320, 1994.

[53] D. Lee and M. Yannakakis. Principles and methods of testing finite state machines - a survey. In *Proc. of IEEE*, volume 84, August 1996.

[54] D. Liu. Reliability and quality assurance of wireless network software. In *Master of Science, Faculty of Graduate Studies*, 2002.

[55] G. Luo, G.V. Bochmann, and A. Petrenko. Test selection based on communicating non deterministic machines using a generalized wp-method. In *IEEE Transactions Software Engineering*, pages 149–162, 1994.

[56] N.A. Lynch and M.R. Tuttle. An introduction to input output automata. In *MIT Report MIT/LCS/TM-373.Cambridge, Massachusetts*, 1988.

[57] M. Mackaya and R. Castanet. Modeling and testing. location based application in umts networks. In *Proc. of IEEE ConTEL 2003*, June 2003.

[58] M. Mackaya, R. Castanet, and O. Koné. Modeling location based application in umts networks. In *Proc. of ACM MSWIM 2002*, September 2002.

[59] A. Mederreg, M. Mackaya, A. Cavalli, and R. Castanet. Test de services basés sur la localisation. In *GRES'03*, 2003.

[60] J. Myers. *The Art of Software Testing.* John Wiley and Sons, 1979.

[61] R. Nahm, J. Grabowsky, and D. Hogreffe. Test case generation for temporal properties. In *Technical Report IAM, University of Berne*, 1993.

[62] S. Naito and M. Tsunoyama. Fault detection for sequential machines by transition tours. In *IEEE Fault Tolerant Computing Conference*, pages 238–243, 1981.

[63] R. De Nicolas and M. Hennessy. Testing equivalences or processes. In *Theorical Computer Science 34*, pages 83–133, 1984.

[64] K. Pehkonen, H. Holma, I. Keskitalo, E. Nikula, and T. Westman. A performance analysis of tdma and cdma based air interface solutions for umts high bit rate services. In *PIMRC'97*, 1-4 September 1997.

[65] A. Petrenko and al. Modeling basic LOTOS by fsms for conformance testing. In *Proc. IFIP International Symposium on Protocol Specification Testing and Verification, Warsaw, Poland*, 1995.

[66] M. Phalippou. Relations d'implantations et hypothèses de test. In *Thèse de doctorat de l'Université de Bordeaux 1*, 1994.

[67] Milner R. A calculus of communicating systems. In *Lecture Notes in Computer Science 92, Springer-Verlag*, 1980.

[68] O. Kone R. Castanet, C. Chevrier and B. Le Saec. A adaptative test sequence generation method for the users needs. In *Proc. IWPTS, Evry, France*, 1995.

[69] O. Rafiq. Le test de conformité de protocoles : une introduction. In *Réseau et informatique Répartie*, pages 101–142, 1991.

[70] O. Rafiq and R. Castanet. From conformance testing to interoperability testing. In *The 3rd Int. Workshop on Protocol Test Systems*, 1990.

[71] O. Rafiq, R. Castanet, and C. Chraibi. Towards an environment for testing OSI protocols. In *Proc of the International Workshop on Protocol Specification, Testing and Verification*, 1985.

[72] K. Sabnani and A. Dahbura. A protocol test generation procedure. In *Computer Networks and ISDN Systems 15*, pages 285–297, 1988.

[73] D. Sidhu and T. Leung. Formal methods for protocol testing : A detailed study. In *IEEE Trans. on Software Engineering*, 1989.

[74] Technical Specification. Universal mobile telecommunications system (umts), stage 2 functional specification of ue positioning in utran. In *3GPP TS 25.305 version 4.2.0*, 2001.

[75] Technical Specification. Digital cellular telecommunication system phase 2+ (gsm),universal mobile telecommunications system (umts) network architecture. In *3GPP TS 23.002 version 4.4.0*, 2003.

[76] Technical Specification. Universal mobile telecommunications system (umts), functional stage 2 description of location services in umts. In *3GPP TS 23.271 version 3.6.0*, 2003.

[77] Information Processing Systems. Open systems interconnection. In *Estelle (a formal description technique based on extended state transition model), ISO IS 9074*, 1989.

[78] OSI Open Source Interconnection. Information Technology. Open systems interconnection conformance testing methodology and framework - part 1 : General concept - part 2 : Abstract test suite specification - part 3 : The tree ans tabular combined notation (TTCN) - part 4 : Test realization. In *International Standard ISO/IEC 9646/1-8*, 1996.

[79] Telelogic. http ://www.telelogic.com.

[80] Telelogic. Objectgeode toolset, http ://telelogic.com. 2002.

[81] G. Touag and A. Rouger. Generation de tests a partir de specifications lda basee sur une construction partielle de l'arbre d'accessibilite. In *Actes du Colloque Francophone sur l'Ingenierie des protocoles. Rabat, Maroc*, 1996.

[82] J. Tretmans. A formal approach on conformance testing. In *thesis, University of Twente, the Netherlands*, 1992.

[83] J. Tretmans. Conformance testing with labelled transition systems : Implementation relations and test generation. In *Computer Networks ans ISDN Systems 29*, pages 49–79, 1996.

[84] C. Viho. Test de conformité et d'interopérabilité : vers une approche répartie. In *Habilitation à diriger des recherches de l'Université de Rennes 1*, 2001.

[85] S.T. Vuong. On test coverage metrics for communication protocols. In *4th International Workshop on Protocol Testing Systems*, 1991.

[86] T. Walter, I. Schieferdecker, and J. Grabowski. Test architectures for distributed systems : state of the art and beyond. In Testing of Communicating Systems, editor, *Petrenko and Yevtushenko*, volume 11, pages 147–174. IFIP, September 1998.

[87] CCITT SG X. Recommendation Z.100. In *Specification and Description Language SDL*, 1987.

[88] ITU-T Z.500 and Committee Draft. Formal methods in conformance testing. In *JTC1/SC21/WG1/Project 54.1*, 1995.

[89] F. Zaidi. Contribution a la génération de tests pour les composants de services. application aux services de réseaux intelligents. In *Thèse de doctorat de l'Université Evry Val d'Essone*, 2001.