Dynamique de l'environnement : Scénarios, simulations et maquette

Badr Benmammar, Zeina Jrad, Francine Krief, Nader Mbarek

Titre : Dynamique de l'environnement : Scénarios, simulations et maquette	Document version : 1.2

Numéro du projet :	Acronyme du projet : IP-SIG	Titre du projet : IP Signaling

Date de livraison : Juillet 2005	Type* / Sécurité** du livrable : R-PU

* Type : P : Prototype, R : Rapport, D : Démonstrateur, O : Autres
** Sécurité : PU : Publique, CO : Confidentiel, restreint au membre du projet

Responsable et éditeur/auteur : Francine Krief	Organisation : LIPN

Auteurs : Badr Benmammar, Zeina Jrad, Francine Krief, Nader Mbarek

Résumé : Ce livrable traite de la négociation dynamique de SLA/SLS et du maintien des paramètres négociés lors de la mobilité du terminal dans le cadre d'une signalisation NSIS. Il fait suite au livrable 4.1 qui présentait un assistant de négociation, coté terminal utilisateur, permettant de négocier dynamiquement les SLA/SLS avec le/les fournisseurs de service, et plusieurs protocoles conformes à l'environnement NSIS afin de négocier le SLS, et maintenir la qualité de service négociée. Il s'agit dans ce livrable de compléter l'approche retenue en présentant des scénarios d'utilisation, notamment dans le cas du protocole de négociation dynamique de paramètres de SLS, des résultats de simulations, principalement pour le protocole de réservation de ressources à l'avance et une maquette permettant de démontrer la faisabilité de la (re)négociation dynamique de SLA/SLS. Cette dernière comporte un module d'apprentissage et un système multi-agents.

Mots clés : Négociation SLA/SLS, Système Multi Agent, Apprentissage numérique, Signalisation NSIS, SLN NSLP, Mobilité du terminal, MQoS NSLP

Table de matières

Introduction

Ce livrable fait suite au livrable 4.1 qui présentait un assistant de négociation, coté terminal utilisateur, permettant de négocier dynamiquement les SLA/SLS avec le/les fournisseurs de service, et plusieurs protocoles conformes à l'environnement NSIS afin de négocier le SLS et maintenir la qualité de service négociée. Il s'agit dans ce livrable de compléter l'approche retenue en présentant des scénarios d'utilisation, notamment dans le cas du protocole de négociation dynamique de paramètres de SLS, des résultats de simulations, principalement pour le protocole de réservation de ressources à l'avance et une maquette permettant de démontrer la faisabilité de la (re)négociation dynamique de SLA/SLS. Cette dernière comporte un module d'apprentissage et un système multi-agents.

Ce livrable est organisé de la façon suivante :

La première partie de ce document traite de la négociation dynamique de SLA/SLS. Cette négociation est possible grace, d'une part, à un assistant de négociation, placé dans le terminal utilisateur, qui permet de choisir le meilleur fournisseur de services et de négocier dynamiquement les paramètres de QoS et d'autre part, à un protocole de négociation de paramètres de SLS conforme à l'environnement de signalisation NSIS.

Concernant l'assistant de négociation, l'accent est mis plus particulièrement sur la description du module d'apprentissage et l'implémentation de la plateforme multi-agents.

Enfin le protocole de négociation SLN NSLP est détaillé et deux scénarios d'utilisation sont décrits. Il s'agit de la négociation pour un service de visioconférence et de voix sur IP.

La deuxième partie de ce livrable traite de la mobilité du terminal. Deux approches permettant d'améliorer la qualité de service dans un réseau sans fil sont présentées.

La première approche est basée sur la technologie agent qui est utilisée lorsqu'un nouvel utilisateur se présente, ce dernier est alors inconnu du système et la réservation de ressources à l'avance est donc impossible car ses futures localisations sont inconnues. L'approche agent vise durant cette phase à adapter le handover aux besoins de qualité de service de l'utilisateur. Les interactions entre les agents sont modélisées à l'aide de AUML et des réseaux de Pétri, c'est ce dernier outil qui nous a permis de faire une vérification du système.

La deuxième approche permettant d'améliorer la qualité de service dans un réseau sans fil, est une approche protocolaire qui est utilisée lorsque le profil de mobilité de l'utilisateur est connu par le système. Dans ce cas, les futures localisations de l'utilisateur sont déterminées et une réservation de ressources à l'avance, basée sur l'application de signalisation QoS NSLP, peut être faite pour lui. Un modèle mathématique est proposé et les résultats de la simulation à l'aide de MATLAB sont présentés.

I Négociation dynamique de SLA/SLS

1. L'assistant de négociation

Nous proposons d'introduire une assistance dans le terminal de l'utilisateur afin de l'aider, d'une part, à choisir le meilleur fournisseur de services et, d'autre part, à négocier dynamiquement les paramètres de qualité de service (QoS) répondant aux exigences de l'utilisateur et de l'application. Le modèle que nous avons élaboré se base sur trois couches principales qui sont : une couche de gestion de profils, une couche de contrôle et une couche de négociation. Pour réaliser ces tâches, nous avons utilisé un mécanisme d'apprentissage numérique et une plateforme multi-agents principalement pour ses propriétés d'autonomie et d'adaptation.

Le modèle d'assistance proposé est placé sur le terminal entre un client et un fournisseur de service, d'une part, et entre un client et un réseau, d'autre part. La tâche principale que doit accomplir cette interface est la représentation de l'utilisateur. Cette représentation se manifeste sous plusieurs aspects :

- Suivre et analyser le travail de l'utilisateur
- Classer les applications ainsi que leurs besoins
- Prendre en compte les caractéristiques du terminal et des réseaux d'accès
- Trouver des profils de négociation
- Garantir/assurer la QoS
- Remplacer/accompagner l'utilisateur dans la prise de décision
- Négocier dynamiquement les paramètres de SLS
- Choisir le meilleur fournisseur de service

1.1. Specification du modèle

L'interface utilisateur (figure1) est représentée par un modèle organisationnel comportant plusieurs couches ou niveaux et une base de données contenant les différentes politiques, règles et propriétés utilisées par les différentes entités du système :

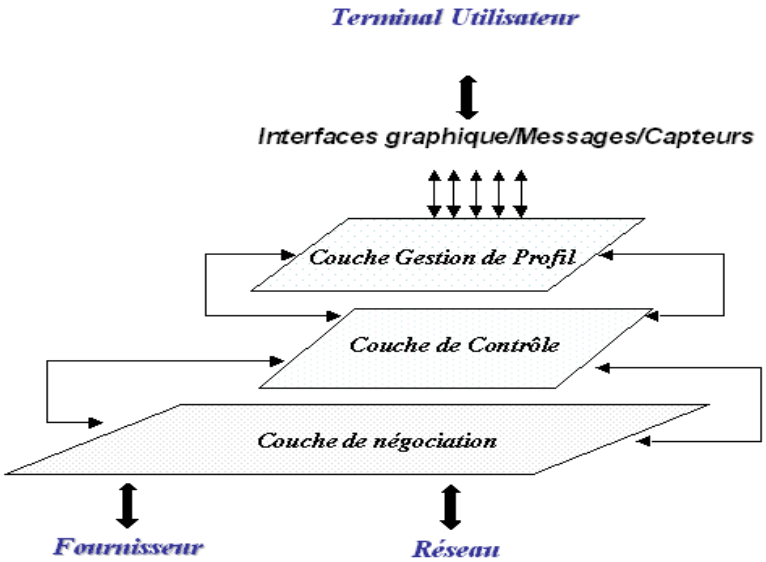


Figure 1. Interface Utilisateur

1.1.1. Couche de gestion de profil

Cette couche est chargée des interactions avec l'utilisateur et de son suivi dans le but d'identifier son profil de négociation et d'être en mesure d'anticiper la réservation des ressources dont il va avoir besoin. Dès sa connexion au terminal, l'utilisateur exécute différents types d'applications (audio, vidéo, client de messagerie, …) qui ont chacune leurs exigences propres en terme de qualité de service. L'interface réagit de façon autonome et mémorise les informations relatives au comportement de l'utilisateur.
Les données prises en compte intègrent une dimension temporelle (heure de lancement de l'application, durée d'utilisation, …) et concernent d'une part les préférences de l'utilisateur (prix, flexibilité, …) et d'autre part les caractéristiques et exigences des applications (débit, délai, taux de pertes, …). Ces informations proviennent essentiellement d'une interview de l'utilisateur à l'aide d'une interface graphique (identité, profession, fréquence d'utilisation, objectifs, …), et du suivi de ses actions (besoins, exigences, …).

Dans la couche de gestion de profil, nous avons adopté les techniques d'apprentissage connexionniste et plus précisément les réseaux de neurones pour apprendre à reconnaître l'utilisateur et l'associer avec un profil de négociation.

1.1.2. L'apprentissage connexionniste

Une carte topologique de Kohonen est un réseau de neurones artificiels dont les paramètres sont adaptés par une procédure d'apprentissage compétitif non supervisé [1]. Elle est formée d'un ensemble de neurones dont les relations de voisinage sont fixées par une « grille » ou topologie du réseau.

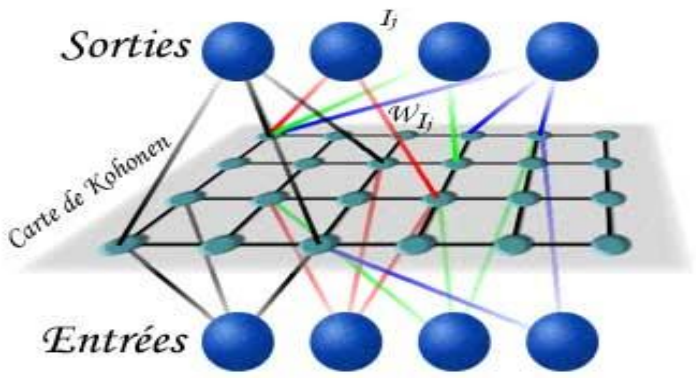


Figure 2. Carte de Kohonen

Chaque neurone dispose d'un référent ou profil qui est dans le même espace que les données. Pendant l'apprentissage, la présentation d'une observation au réseau active le neurone dont le référent est le plus proche. Les référents du neurone « gagnant » et de ses voisins sont alors rapprochés de l'observation. Les unités s'organisent ainsi progressivement en fonction de leur similarité. Elle permet d'une part de visualiser des données multidimensionnelles en préservant leur topologie locale (deux observations proches dans l'espace des données sont projetées sur des neurones proches sur la carte), d'autre part d'identifier les tendances ou régularités présentes dans les données en regroupant les observations similaires en *clusters*. Les neurones ont des espaces de dimensions différents (figure 3), voisinage linéaire, rectangulaire ou exagonal.

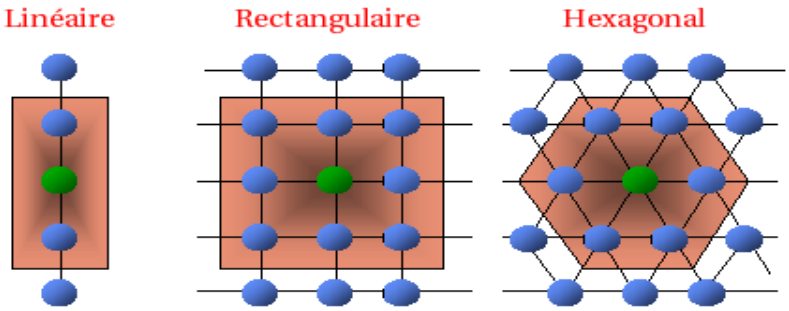


Figure 3. Différents types de voisinage

1.1.2.1. Première approche

Notre approche consiste à intégrer un module d'apprentissage qui récupère des fichiers log qui sont les traces de navigation ou d'utilisation récupérées du serveur/système [9] [10]. Ces fichiers log seront nettoyés et recodés au format numérique ou binaire pour qu'ils soient utilisables par l'algorithme d'apprentissage et les cartes de kohonen, le but étant de construire une carte topologique à partir du fichier recodé dans le but d'extraire des profils (figure 4).

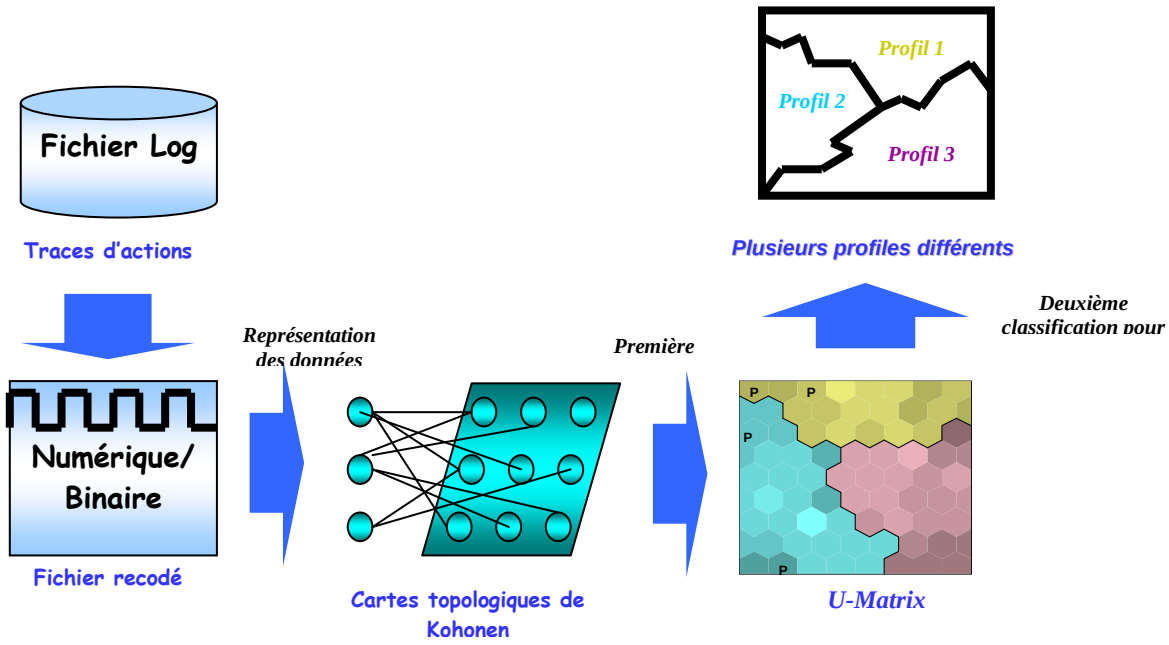


Figure 4. Apprentissage connexionniste

Les différentes étapes de cet apprentissage consistent à :

1. Premièrement, créer une base de données avec les différents scénarios ou contextes d'utilisations qui contiennent à la fois les caractéristiques de l'utilisateur, les besoins des applications, les caractéristiques du terminal et les informations de retour sur la satisfaction de l'utilisateur.

Ces données sont illustrées dans l'exemple suivant :

Numéro d'identification ou de session	Login de l'utilisateur	Intitulé de l'application	Intitulé de l'application	Intitulé de l'application	Exécution			Descriptif	QoS attribuée
					Date	Heure de Départ	Heure d'Arrêt		
8898	Zj	Vic			12/10/1999	08 : 06 : 16	09 : 06 : 16	Professionnel	
8898	Zj	Vic	Données		12/10/1999	08 : 16 : 00	09 : 06 : 16	Professionnel	
8898	Zj			Audio	12/10/1999	08 : 30 : 16	09 : 06 : 16	Loisir	
1999	kml	Rat			16/11/1999	08 : 14 : 58	09 : 06 : 16	Normal	
15654	vhg	e-mail			16/11/1999	08 : 21 : 34	09 : 03 : 19	----------	----
15654	vhg	e-mail	Vidéo		16/11/1999	08 : 21 : 34	09 : 03 : 19	----------	----

Dans le tableau ci-dessus, une ligne correspond à un état et une séquence est une transition d'états. Le numéro d'identification peut être unique pour chaque session ou chacun des contextes d'utilisation, ceci est un moyen supplémentaire et relativement simple permettant de distinguer rapidement le travail de l'utilisateur. Le Numéro d'Identification correspond à la clé primaire de la base de données, ce qui est mieux que la date et l'heure qui peuvent correspondre à plusieurs applications lancées en même temps.

2. Traitement des données pour construire des SOM (self organizing map) à partir de cette base dans le but de séparer les différentes catégories de contextes, c'est-à-dire trouver les profils types. Le Self Organizing Map (SOM) de Kohonen permet à la fois de classer des objets et de visualiser le résultat de cette classification tout en donnant une indication sur les "positions relatives" des classes. Le SOM est limité aux données numériques avec une possibilité d'extension du SOM aux données non numériques.
   Les SOM mettent en oeuvre un processus d'organisation spatial selon lequel les unités de input sont classées selon leur caractéristiques. Le réseau partage logiquement l'espace de input en "cluster", où prévalent les composantes principales qui différencient les données en entrée. Les SOM sont des réseaux qui évoluent avec un apprentissage non supervisé et auto organisé, en faisant fonctionner les unités de l'input en compétition selon un principe de " winner unit ". Les résultats des SOM ont permis de réaliser plans de " clustering " (figure 5) où les zones sont identifiées soit par leur fortes similarités soit par la position qu'elles occupent dans la matrice de l'output (pour laquelle la dimension doit être définie au début du processus neural).

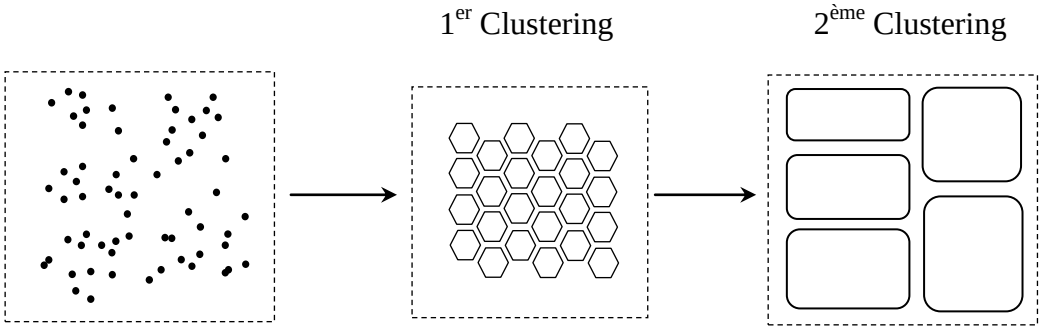


Figure 5. Segmentation des SOM

Les SOM donnent beaucoup de classes car chaque neurone représente une classe. Ce qui fait qu'on a besoin d'un deuxième clustering qui consiste à chercher des groupes d'indices homogènes ou chercher des clusters bien séparés. Avec les SOM, on a identifier des groupes et des relations de proximité entre les groupes, seulement ces groupes sont trop nombreux pour être analysés 1 à 1, donc on fait un deuxième découpage à l'aide de l'algorithme de K-means.
Le K-means est un algorithme de classification automatique. Tout seul, on peut obtenir des profils mais pas de possibilité de visualisation, c'est pour cela qu'on utilise les SOM et K-means. K-means clustering impose de fixer le nombre de clusters ou de classes et fait varier le nombre de classes de 1, 2, 3, 4… Pour chacun de ces nombres, on calcule l'indice, le nombre de clusters pour la valeur minimale de l'indice sera optimal et c'est cette valeur que nous retiendrons

3. Prédiction du profil (figure 6) dans le but d'anticiper la négociation. Ceci revient à associer, en premier lieu, chaque état ou contexte avec un neurone ID. Ensuite, il s'agit de construire une deuxième base de données qui représente la relation entre les états et les neurones ID, déterminer à partir du contexte actuel et de la base de données le futur contexte, le prochain neurone ID et donc le prochain profil. Il se peut que le prochain neurone ID appartienne à la même classe ; dans ce cas, il n'y aura pas de changement de profil de négociation. Donc deux cas de figure se présentent : un utilisateur qui, durant la même session, change de profil ; et un utilisateur qui, durant la même session garde le même profil tout au long de son travail. De cette façon, nous pouvons dire que le système est adaptatif car il prend en compte la modification des besoins de l'utilisateur durant une même session.

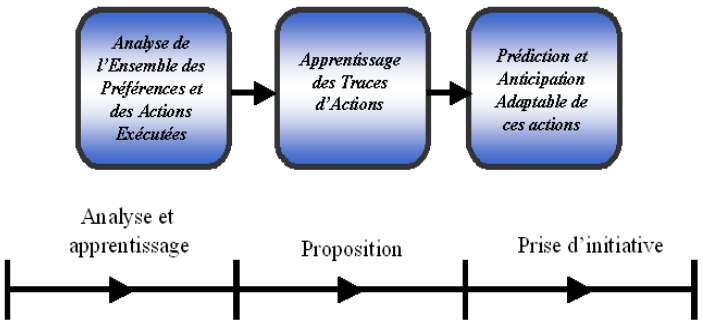


Figure 6 : Prédiction du comportement

Illustration : Prenons par exemple la première session du tableau ci-dessus. La première ligne correspond à un état/contexte. Cet état représente le lancement de l'application Vic par zj. Après la construction des SOM et la différenciation des profils types, cet état sera projeté sur un neurone qui a un ID bien précis. Il suffit de trouver à quelle classe appartient ce neurone pour trouver le profil actuel de la session.

1.1.2.2. Amélioration du modèle, deuxième approche

Le nombre de facteurs qui influencent le comportement d'un utilisateur est trop important pour établir manuellement une politique de négociation de ressources. Par ailleurs, le système doit s'adapter à l'utilisateur et à ses évolutions comportementales. Notre approche intègre donc un module d'apprentissage capable de s'adapter à de nouveaux comportements (plasticité) sans oublier les comportements antérieurs (stabilité).
Nous utilisons le modèle M-SOM-ART [2] développé à l'origine pour la classification des utilisateurs d'un site de commerce en ligne en deux catégories : acheteurs et non-acheteurs. Ce modèle est une adaptation des cartes topologiques de Kohonen (SOM) [4] adaptées aux traitements des séquences (M-SOM) [3] plongées dans le paradigme de la résonance adaptative (ART) [5]. Il dispose de toutes les propriétés nécessaires: plasticité, stabilité et prise en compte de la dimension temporelle.

Le modèle M-SOM-ART permet de prendre en compte d'une part la nature temporelle des données et il intègre d'autre part les propriétés de stabilité et de plasticité nécessaires à un apprentissage continu. En effet, des neurones sont ajoutés au réseau lorsqu'une nouvelle observation est trop éloignée du référent le plus proche de la carte et dont l'adaptation aurait conduit à l'oubli des connaissances acquises antérieurement.

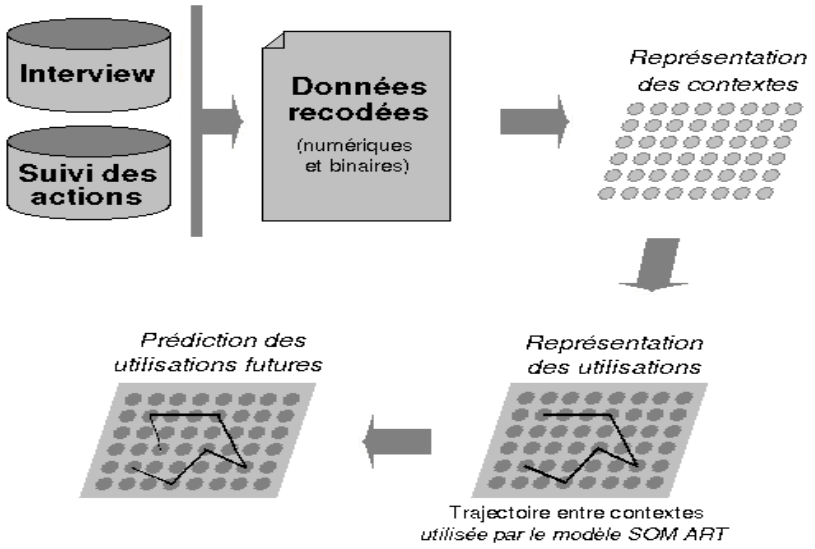


Figure 6 : Processus d'apprentissage numérique

Une carte de Kohonen est construite pour représenter les contextes d'utilisation (préférences utilisateur, exigences de l'application, …). Les sessions sont alors représentées par des trajectoires sur la carte des contextes qui permettent de construire le modèle M-SOM-ART des comportements. Ce dernier modèle permet d'identifier le profil de l'utilisateur à partir des actions qu'il a effectué depuis le début de sa session et de prédire les applications qu'il va exécuter. Le système dispose alors d'informations nouvelles qui lui permettent d'anticiper la réservation de ressources.

1.2. Réalisation

La réalisation de l'interface utilisateur décrite ci-dessus et présentée dans le livrable 4.1 se base sur deux parties principales qui sont :

1. L'application des mécanismes d'apprentissage numérique sur des données qui représentent des contextes d'utilisations variés dans le but de séparer les différents comportements types et de pouvoir associer chaque utilisation avec un profil type.
2. L'utilisation d'une plateforme multi-agents pour l'implémentation du processus de négociation entre les différents agents du modèle.

1.2.1. Les données contextuelles

Le tableau ci-dessous contient les paramètres retenus pour décrire des informations personnelles et générales sur l'utilisateur, des paramètres qui décrivent des informations sur les préférences d'un utilisateur et des informations propres à chaque application utilisée :

Variable ID	Paramètre	Description	Type
1	**Langages_FR**	Langue parlée par l'utilisateur	*Binaire {0/1}*
2	**Jour_Semaine**	Jour de la connexion	*Binaire {0/1}*
3	**Samedi**	Jour de week-end	*Binaire {0/1}*
4	**Dimanche**	Jour de week-end	*Binaire {0/1}*
5	**Heure**	Start time of the connexion	*Numerique[0-24]*
6	**Durée_Connexion**	Durée de la connexion totale en nombre d'heures	*Numerique[0-10]*
7	**Connaissance**	Niveau de connaissances dans le domaine	*Numerique[0-10]*
8	**Importance**	Le degré d'importance du travail effectué globalement	***Numerique[0-10]***
9	**Appli 1**	Id de la première application lancée	*Binaire{0/1}*
10	**Appli 2**	Id de la deuxième application lancée	*Binaire{0/1}*
11	**Appli 3**	Id de la première application lancée	*Binaire{0/1}*
12	**Appli 4**	Id de la première application lancée	*Binaire{0/1}*
13	**From**	Pour chaque application lancée	*Numerique[0-24]*
14	**Durée**	Pour chaque application lancée	*Numerique[0-10]*
15	**Activité courante**	Loisir/ Travail	*Binaire{0/1}*
16	**Exigences Quality**	Les exigences sur la qualité:Le niveau de la qualité demandée	*Numerique[0-10]*
17	**Exigences Prix**	Les exigences sur le prix	*Numerique[0-5]*
18	**Satisfaction**	De l'utilisateur pour cette application	*Numerique[0-5]*

Nous avons pris en considération dans notre implémentation trois types d'utilisation majeure à savoir :

1. Utilisateurs qui travaillent dans la semaines [Lundi - Vendredi], dans la journée [8h – 18h], qui restent connectés pour une longue durée [> 4h]. Ils parlent plusieurs langues, ont un niveau de connaissance assez élevé [>7], accordent beaucoup d'importance à leur travail [>7], où l'activité courante est la plupart du temps professionnelle, et qui sont très exigeants sur la qualité [>7] et très peu exigeants sur le prix [<4]. Ce sont des utilisateurs qui lancent des applications de type 3 [entre 8h et 18h] sur lesquelles ils travaillent pendant une durée maximale de 2h interrompue pour le déjeuner [12h - 14h]. Les applications Type3 nécessitent des besoins en Delai compris entre Dmin et Dmax, une Gigue entre Gmin et Gmax, une Perte entre Pmin et Pmax et une Bandpassante entre Bmin et Bmax. Ils lancent aussi des applications de Type 2 entre 14h et 16h et restent connectés au maximum 30 minutes. Les applications Type 2 nécessitent des besoins en Delai compris entre Dmin et Dmax, une Gigue entre Gmin et Gmax, une Perte entre Pmin et Pmax et une Bandpassante entre Bmin et Bmax.

2. Utilisateurs qui travaillent le week-end [Samedi & Dimanche], dans la soirée [19h – 23h], qui restent connectés pour une courte durée [< 3h]. Ils parlent une seule langue, ont un niveau de connaissance très faible [<4], accordent très peu d'importance à leur travail [<4], où l'activité courante est la plupart du temps liée au loisir, et qui sont très exigeants sur la qualité [>7] et très exigeants sur le prix [<4]. Ce sont des utilisateurs qui lancent des applications de Type 2 [entre 19h et 18h] sur lesquelles ils travaillent pendant une durée maximale de 30 minutes. Les applications Type 2 nécessitent des besoins en Delai compris entre Dmin et Dmax, une Gigue entre Gmin et Gmax, une Perte entre Pmin et Pmax et une Bandpassante entre Bmin et Bmax. Ils lancent ensuite des applications de Type 1 et restent connectés au minimum 2 heures. Les applications Type1 nécessitent des besoins en Delai compris entre Dmin et Dmax, une Gigue entre Gmin et Gmax, une Perte entre Pmin et Pmax et une Bandpassante entre Bmin et Bmax.

3. Utilisateurs qui travaillent dans la semaines [Lundi - Vendredi], dans la journée [8h – 18h], qui restent connectés pour une moyenne durée [entre 3 et 6 heures]. Ils parlent deux langues, ont un niveau de connaissance moyen [4-7], accordent une moyenne importance à leur travail [4-7], où l'activité courante est la plupart du temps professionnelle, et qui sont moyennement exigeants sur la qualité [4-7] et moyennement exigeants sur le prix [4-7]. Ce sont des utilisateurs qui lancent que des applications de type3 sur lesquelles il travaillent pendant une durée [2-4h] interrompue pour le déjeuner [12h - 14h]. Les applications Type3 nécessitent des besoins en Delai compris entre Dmin et Dmax, une Gigue entre Gmin et Gmax, une Perte entre Pmin et Pmax et une Bandpassante entre Bmin et Bmax

Dans le but de pouvoir simuler des scénarios d'utilisation et de pouvoir montrer l'application de l'apprentissage et la gestion des profils utilisateur, nous avons généré des données qui représentent plusieurs cas de figure réel. La figure ci-dessous montre une partie du code pour la génération des données avec une partie des données résultantes. La première ligne du tableau représente les variables décrites dans le tableau ci-dessus.

```
% SESSIONS DATA
for session = 1 : nb_session
  M = rand(Temps(session,5),size(Cappli,2));
  M = M -  (ones(Temps(session,5),1) *  mean(M));
  M = M ./(ones(Temps(session,5),1) *   std(M));
  Appli = real(M * Cappli^0.5);
  Appli = Appli -  (ones(Temps(session,5),1) *  min(Appli));
  Appli = Appli ./(ones(Temps(session,5),1) *  max(Appli));
  % partie QOS (delai,  gigue, perte, bandwidth)
  Appli(:,1:4) = Appli(:,1:4)  .* (ones(Temps(session,5),1) * delta_appli3);
  Appli(:,1:4) = Appli(:,1:4) +  (ones(Temps(session,5),1) * min_appli3);
  Appli(:,1:4) = round(Appli(:,1:4));
  % partie heure de debut
  Appli(:,5) = Appli(:,5)  .* (ones(Temps(session,5),1) ...
                  * [ Temps(session,2) - Temps(session,1) ...
                              - Temps(session,4) + Temps(session,3) ]);
  Appli(:,5) = Appli(:,5) +  (ones(Temps(session,5),1) * Temps(session,1));
  Appli(:,5) = Appli(:,5) +  ( (Appli(:,5) > Temps(session,3)) .* ...
                    (ones(Temps(session,5),1) ...
                *(Temps(session,4) - Temps(session,3))));
  % partie duree
  Appli(:,6) = Appli(:,6)  .* (ones(Temps(session,5),1) ...
                  * [ d_appli3_max - d_appli3_min]);
  Appli(:,6) = Appli(:,6) +  (ones(Temps(session,5),1) * d_appli3_min);
  % tri selon l'heure de debut et la duree
  Appli        = sortrows(Appli,[5 6]);
  [HeureLiber I] = sortrows((Appli(:,5) + Appli(:,6)));
  %
  % calcul des besoins cumules
  Ctxt = [];
  lastLib   = 0;
  hLastChgt = Temps(session,1); % heure du dernier changement
    % libération des ressources
    while (   (lastLib < Temps(session,5)) & (HeureLiber(lastLib + 1) < Appli(chgt,5)) )
      lastLib = lastLib + 1;
      Ctxt = [ Ctxt; max(Appli(1:chgt,1:4)), ...
            HeureLiber(lastLib,1) - hLastChgt ];
      hLastChgt = HeureLiber(lastLib,1);
    end
    % calcul du cumul des ressources allouees
    Ctxt = [ Ctxt; max([0,0,0,0;Appli(1:chgt,1:4)]), ...
          Appli(chgt,5) - hLastChgt ];
    hLastChgt = Appli(chgt,5);
  end
  data=[data; ones(size(Ctxt,1),1)*[User(session,:) Day(session,:) ] Ctxt];
end
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
2474	1	1	8	10	1	9	1	0	1	0	0	0	0	0	3	5	7	6	0
2475	1	1	8	10	1	9	1	0	1	0	0	0	0	0	3	5	7	8	3.3995
2476	1	1	8	10	1	9	1	0	1	0	0	0	0	0	3	5	7	8	0.1945
2477	1	1	8	10	1	9	1	0	1	0	0	0	0	0	3	5	7	8	0.37959
2478	1	1	8	10	1	9	1	0	1	0	0	0	0	0	3	5	7	8	0.12923
2479	1	1	8	10	1	9	1	0	1	0	0	0	0	0	3	7	7	8	0.19885
2480	1	1	8	10	1	9	1	0	1	0	0	0	0	0	3	7	7	8	1.9093
2481	1	1	8	10	1	9	1	0	1	0	0	0	0	0	3	7	7	5	0.58614
2482	1	1	8	10	1	9	1	0	1	0	0	0	0	0	3	7	7	5	0.26875
2483	1	1	8	10	1	9	1	0	1	0	0	0	0	0	3	7	7	5	0.2838
2484	1	1	8	10	1	9	1	0	1	0	0	0	0	0	3	7	7	5	0.94112
2485	1	1	8	10	1	9	1	0	1	0	0	0	0	0	3	7	9	8	0.77801
2486	1	1	8	10	1	9	1	0	1	0	0	0	0	0	3	7	9	8	1.198
2487	1	1	8	10	1	9	1	0	1	0	0	0	0	0	3	7	9	8	0.74545
2488	0	1	8	9	1	8	3	1	0	0	0	0	0	0	3	6	7	8	0
2489	0	1	8	9	1	8	3	1	0	0	0	0	0	0	3	7	7	8	0.22119
2490	0	1	8	9	1	8	3	1	0	0	0	0	0	0	3	7	9	8	1.7933
2491	0	1	8	9	1	8	3	1	0	0	0	0	0	0	3	7	9	8	1.494
2492	0	1	8	9	1	8	3	1	0	0	0	0	0	0	3	7	7	8	0.91661
2493	0	1	8	9	1	8	3	1	0	0	0	0	0	0	3	7	7	8	0.6514

1.2.2. Négociation multi-agents

Pour des propriétés telles que l'adaptation, l'autonomie [Ferber] et la coopération, nous proposons d'implémenter cette interface à l'aide d'un système multi-agents. Ce système multi-agent est constitué de plusieurs agents autonomes qui peuvent communiquer entre eux pour échanger des informations ou demander des services [6].

Nous avons tout d'abord étudié les principales plateformes multi-agents disponibles aujourd'hui (voir annexe). Nous avons ainsi pu constater que peu d'efforts ont été consacrés à la standardisation des plateformes SMA. Seuls quelques groupes de recherche et d'industriels (OMG, FIPA, KAOS, General Magic Group) ont commencé ce processus de standardisation. De plus, il n'existe pas d'environnement ou de plateforme multi-agents complet. Un bon environnement doit à la fois offrir de l'assistance à l'utilisateur comme c'est le cas dans la platefome MERCURE, être ouvert voire auto-organisé comme dans MACE, intégrer l'hétérogénéité comme dans MASK et DIMA, fournir des outils de suivi comme MAST ou de simulation dans CORMAS, permettre de valider un système, même partiellement, par rapport aux spécifications requises et offrir des capacités de déploiement comme dans Vulcano.

Pour implémenter notre système, nous avons retenu la plate forme « JADE-Java Agent Development Framework ». JADE est un middleware développé par TILAB pour le développement de systèmes multi agents distribués.

Jade (pour Java Agent DEvelopment framework) est un outil qui répond aux normes de la FlPA. C'est un middleware écrit en Java et se conformant aux spécifications de la FIPA. Cet environnement simplifie le développement d'agent en fournissant les services de base définis par la FIPA, ainsi qu'un ensemble d'outils pour le déploiement. L'outil possède trois modules principaux (nécessaires aux normes PIPA) : le DF « director facilitor » fournit un service de

pages jaunes à la plateforme ; le ACC « agent communication chanel » gère la communication entre les agents ; le AMS « agent management system » supervise l'enregistrement des agents, leur authentification, leur accès et utilisation du système. Les agents communiquent par le langage FIPA ACL. Un éditeur est disponible pour l'enregistrement et la gestion des agents. Aucune autre interface n'est disponible pour le développement ou l'implémentation. À cause de cette lacune, l'implémentation demande beaucoup d'efforts. Elle nécessite une bonne connaissance des classes et des différents services offerts.

Une plateforme Jade [7] peut être répartie sur un ensemble de machines et configurée à distance. Cependant, la configuration du système peut être altérée dynamiquement, à cause d'un mécanisme de migration d'agents au sein de la même plateforme [8].

Jade est basé sur une architecture de communication de type Peer to Peer. De nombreuses raisons nous ont poussé à choisir la plate forme Jade :

- Elle est compatible FIPA
- La communauté utilisatrice est de plus en plus importante
- Elle permet l'exécution distribuée
- Elle permet l'exécution concurrente des agents
- La communication est transparente : message (ACL)
- Elle propose la notion de services
- Elle a été testée sur des terminaux mobiles
- Elle est relativement facile à utiliser
- Elle permet la migration des agents

Nous pouvons décomposer cette phase de réalisation en deux sous phases : une phase de mise en place de l'architecture c'est à dire du protocole et une phase d'intégration des modules de décision des agents et des objets SLS.
La première phase a consisté à mettre en place les agents qui participent aux interactions. Il s'agit des agents cotés utilisateurs (coordinateur, négociateurs) et des agents cotés fournisseurs (coordinateur et négociateurs). Nous créons un agent coordinateur qui crée à son tour dynamiquement 2 agents négociateurs, récupère les adresses de plateformes fournisseurs et entame une négociation avec des agents situés sur cette plate forme.
A ce stade, les agents ne disposent pas encore de module de décision. Le protocole de négociation retenu est ensuite implanté dans chaque agent.

Environnements :

L'environnement de négociation représente l'ensemble des facteurs qui influent sur le résultat de la négociation. Dans notre cas, il est caractérisé par le nombre d'agents, les paramètres sur lesquels porte la négociation, les temps impartis pour la négociation (deadline) et les attentes des agents (en terme de bénéfice). Le nombre d'environnements possibles est infini. Afin d'évaluer les performances des négociations, il est nécessaire de se restreindre à un sous ensemble de ces environnements Dans la première série de simulations, les négociations porterons sur un paramètre (le prix) et opposeront principalement deux agents : un fournisseur et un utilisateur, ainsi que leurs agents coordinateurs.
Lors d'une simulation, on distingue les variables indépendantes et les variables dépendantes. Les variables indépendantes sont les variables contrôlées (fixés) par celui qui mènent les simulations. Les variables dépendantes sont les variables observées, résultats des expérimentations.
Les variables indépendantes sont les suivantes :
- Ka : La position relative de la première offre par rapport aux valeurs limites

- Les intervalles de négociation [min, max](client) et [min, max](serveur) pour le paramètre.
- Tmax : le temps maximal de négociation (deadline).
- Beta : le degré de concession d'un agent au cours du temps

TEST DES DIFFERENTES ETAPES DU PROTOCOLE AU COURS DE LA NEGOCIATION

Le temps est exprimé en millisecondes.

Client :

L'intervalle de négociation pour le fournisseur est [100.0, 800.0]
Il dispose de 2000 ms pour négocier, la valeur de sa première offre est 0.15 fois la longueur de l'intervalle de négociation. Son degré de concession Beta = 2.0 (il est assez proche d'une concession linéaire)
prixlMin : 100.0
prixMax : 800.0
tempsMax : 2000
firsKa : 0.15
beta :2.0

Fournisseur :

L'intervalle de négociation pour le fournisseur est [300.0, 1500.0]. Il dispose de 2000 ms pour négocier, la valeur de sa première offre est 0.15 fois la longueur de l'intervalle de négociation. Son degré de concession Beta = 4.0 (il est assez proche d'une concession linéaire, mais concède plus facilement que le client) :
prixMin : 300.0
prixMax : 1500.0
tempsMax : 2000
firsKa :0.15
beta :4.0

1. Agent utilisateur : Envoie du message CFP de l'agent négociateur de l'utilisateur à l'agent négociateur du fournisseur.

Ce message marque le début de la négociation. L'agent négociateur de l'utilisateur *(sender : buyer1@machiavel7:1099/JADE)* envoie un message CFP à l'agent négociateur du fournisseur
(receiver : seller1@machiavel7:1099/JADE).
Le contenu du message est un objet Offre qui contient le nom du service qu'il veut négocier.

2. Agent Fournisseur : Réception d'un message CFP par le fournisseur et envoi d'une première offre : message « PROPOSE »

L'agent négociateur du fournisseur (seller1) reçoit le message CFP, il lit le nom del'article recherché (vidéo) et génère en fonction des paramètres affichés
(temps :70, tempsMax : 2000, firsKa :0.15, beta :4.0)
une valeur pour la fonction polynomiale alpha(t).
Il génère ensuite à partir de alpha(t), la valeur qu'il s'apprête à proposer :
878.8186118386448 et envoie un message « PROPOSE » à l'agent (buyer1) contenant cette valeur.

3. Agent utilisateur : Réception d'un message "PROPOSE" par le client, traitement et envoi d'un message « PROPOSE »

L'agent de l'utilisateur (buyer1) reçoit un message « PROPOSE » contenant une proposition

sur le prix de l'article qu'il veut acquérir.
le prix du vendeur : seller1@machiavel7:1099/JADE est : 878.8186118386448.
A l'aide de son module de décision, il génère lui même une valeur :
calcul génération polynomiale min :378.98027905483997
et vérifie si l'utilité de son offre est supérieure à celle qui lui est proposé. Dans notre cas, il refuse l'offre et propose la valeur qu'il vient de générer :
proposition refusée, nouvelle proposition
proposition : 378.98027905483997
Le client renvoie un message « PROPOSE » contenant la valeur générée à l'agent fournisseur seller1.

4. Agent utilisateur : Réception d'un message "PROPOSE" par le client, traitement et envoi d'un message « PROPOSE ».
A la suite de plusieurs échanges de « PROPOSE » entre l'agent fournisseur et l'agent utilisateur, la proposition de ce dernier est acceptée par l'agent fournisseur. Nous montrons la génération de cette proposition par l'agent utilisateur.
L'agent utilisateur reçoit une proposition :
le prix du vendeur : seller1@machiavel7:1099/JADE est : 594.8165270739671
il refuse l'offre et propose à son tour :
proposition refusée, nouvelle proposition
proposition : 547.0603199729544
On s'aperçoit que contrairement au début des échanges, les valeurs proposées sont assez proches.

5. Agent fournisseur : Réception d'un message PROPOSE, envoi d'un message "ACCEPT_PROPOSAL"
L'agent négociateur du fournisseur reçoit la proposition
le prix proposé par le client : buyer1@machiavel7:1099/JADE est : 547.0603199729
Il génére une nouvelle valeur à l'aide de son module de décision
calcul generation polynomiale max :524.2122952346665
il se rend compte que l'utilité de l'offre reçue est supérieure à l'utilité de l'offre qu'il s'apprête à envoyer. Il accepte alors la proposition. Il s'est écoulé 741 ms :
temps ecoulÚ :741
temps max :2000
accept
seller1 SEND ACCEPT_PROPOSAL MESSAGE

6 . Agent fournisseur: envoi d'un message AGREE à son coordinateur
Apres avoir envoyé un message « ACCEPT » à l'agent utilisateur, l'agent fournisseur envoie un message AGREE a son agent coordinateur afin de confirmer la disponibilité du service:
seller1 SENT AGREE MESSAGE : (AGREE
:sender (agent-identifier :name seller1@machiavel7:1099/JADE)
:receiver (set (agent-identifier :name provider1@machiavel7:1099/JADE :addresses

7. Agent coordinateur fournisseur : Réception d'un message AGREE, et envoi d'un message "CONFIRM"
L'agent coordinateur (provider1) du fournisseur reçoit un message « AGREE » de l'un de ses agents négociateurs.
provider1 GET AGREE MESSAGE : (AGREE

:sender (agent-identifier :name seller1@machiavel7:1099/JADE)
Il vérifie la disponibilité du service et envoie un message de confirmation à l'agent «(seller1) :
provider1 SEND CONFIRM MESSAGE : (CONFIRM
:receiver (set (agent-identifier :name seller1@machiavel7:1099/JADE))

8 . Agent fournisseur : Réception du message « CONFIRM » et envoi d'un message « INFORM »
L'agent fournisseur reçoit de la part de son agent coordinateur la confirmation que le service est disponible. Il envoie alors un message « INFORM » à l'agent utilisateur pour lui signaler qu'il est prêt à lui fournir le service spécifié.

9. Agent utilisateur : Réception d'un message "INFORM" et envoi d'un message « AGREE »
L'agent utilisateur a reçu un message du fournisseur lui spécifiant que sa dernière offre a été acceptée et que l'agent fournisseur vérifie la disponibilité du service. Il attend un message « INFORM » lui confirmant cette disponibilité.
Dés réception de ce message :
buyer1 GET INFORM MESSAGE : (INFORM
:sender (agent-identifier :name seller1@machiavel7:1099/JADE)
:receiver (set (agent-identifier :name buyer1@machiavel7:1099/JADE))
:content "547.0603199729544"
Il envoie par un message « AGREE » le résultat de la négociation à son agent coordinateur.
:receiver (set (agent-identifier :name coordinator@machiavel7:1099/JADE)

10. Agent coordinateur utilisateur : Réception d'un message "AGREE" et envoi d'un message « CONFIRM »
L'agent coordinateur récupère les messages « AGREE » des différents threads de négociation (dans notre cas un thread) , les classe suivant leur utilité et envoie un message « CONFIRM » à l'agent qui a envoyé l'offre la plus intéressante.

11. Agent négociateur : Réception d'un message « CONFIRM » et envoi d'un message « INFORM »
L'agent négociateur qui a envoyé l'offre la plus intéressante reçoit un message « CONFIRM » de la part de son agent coordinateur. Il envoie un message « INFORM » à l'agent du fournisseur pour lui confirmer/valider le préaccord établi sur le service. Il s'engage à utiliser le service.

12. Agent fournisseur : Réception d'un message « INFORM »
L'agent négociateur du fournisseur récupère un message « INFORM » venant de l'agent de l'utilisateur qui signifie que celui-ci confirme le préaccord qu'ils ont pris sur le service et qu'il est prêt à utiliser le service.

1.2.3. Renégociation à la demande du fournisseur de service

Le but de cette maquette (figure 7) est de montrer l'intérêt d'une renégociation dynamique pour le fournisseur de service. Ce dernier, lorsque son réseau commence à être trop chargé peut augmenter dynamiquement le prix de ses services. Au contraire, lorsque son réseau ne n'est pas suffisamment utilisé, il peut proposer une baisse de ses tarifs. La possibilité de renégocier

dynamiquement les prix lui permet ainsi d'optimiser l'utilisation de ses ressources et d'éviter toute dégradation de service.

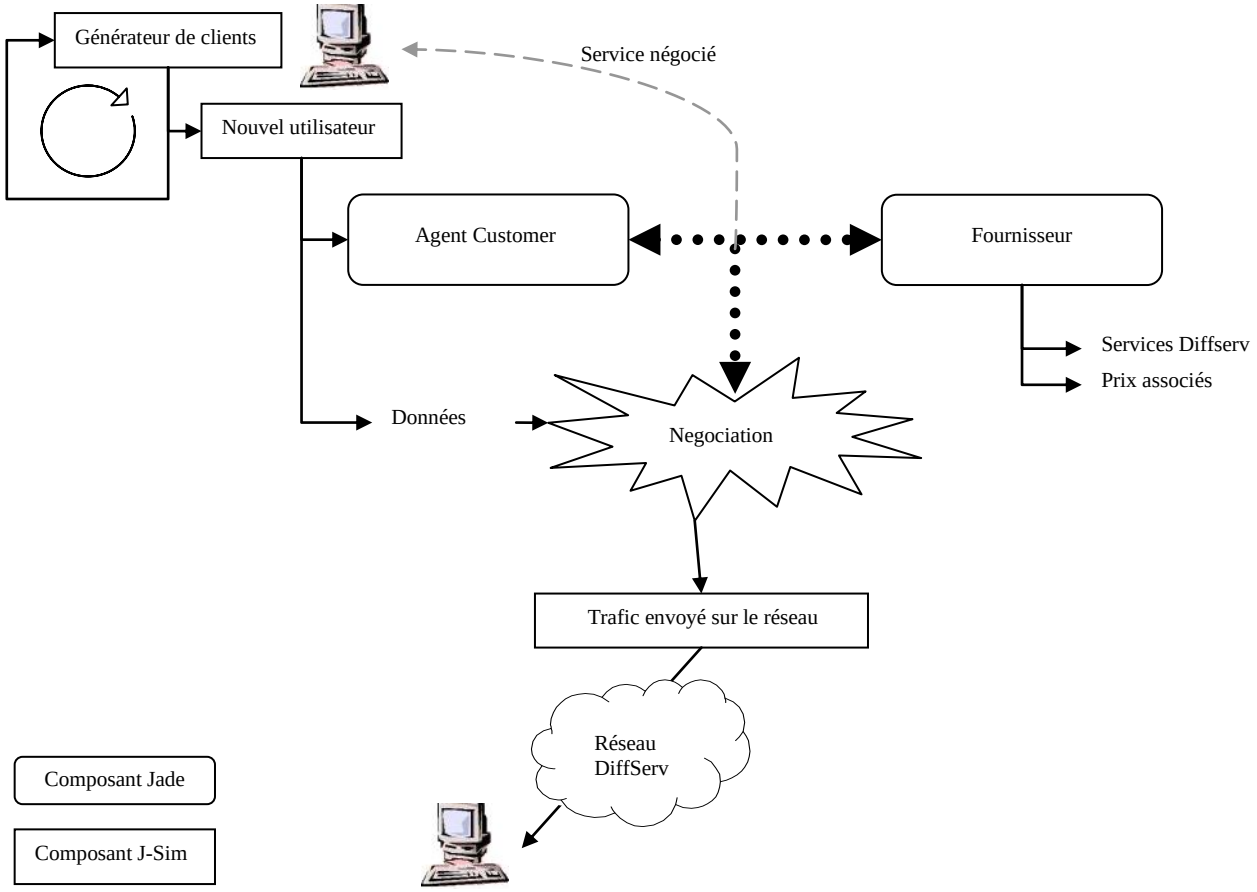
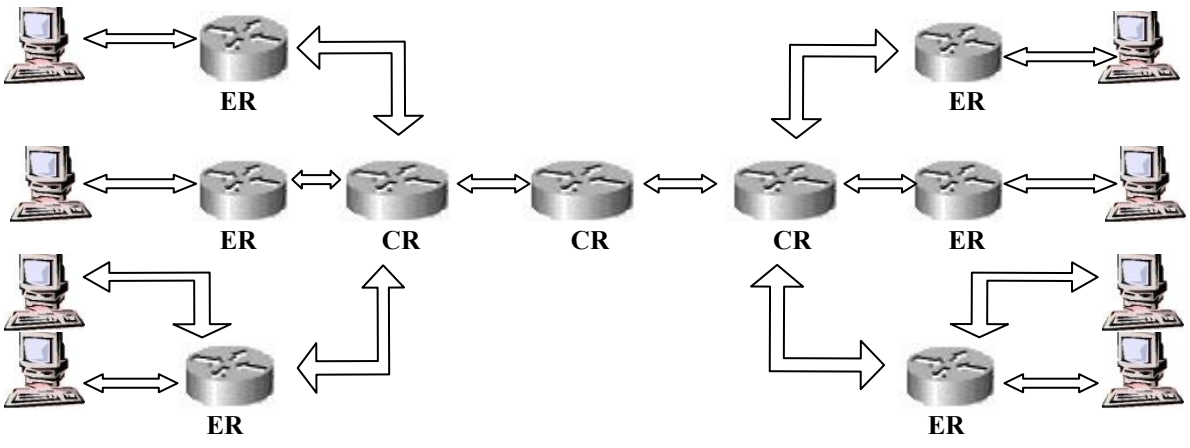


Figure 7 : Renégociation suite à une demande du fournisseur de service

Le réseau est simulé à l'aide du logiciel J-Sim (Java Simulator) (figure 8).



ER: Edge Router

CR: Core Router

Figure 8. Réseau simulé

La génération du trafic se fait en respectant la répartition suivante : 33% Gold – 37% Silver – 5% Bronze – 25% Best Effort.

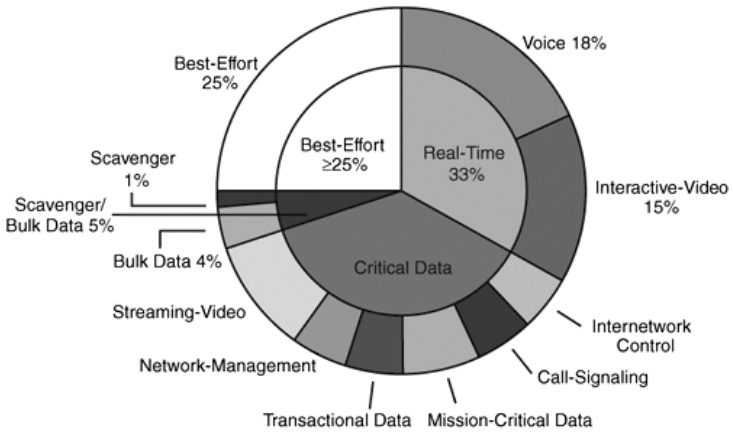


Figure 9. Répartition du trafic

1.3. Références

[1] [1] T. Kohonen, (2001), "Self-Organizing Maps“ Springer Series in Information Sciences, Vol. 30, Springer, Berlin, Heidelberg, New York, 1995, 1997, 2001. Third Extended Edition, 501 pages. ISBN 3-540-67921-9, ISSN 0720-678X.

[2] F. Zeharoui and Y. Bennani, M-SOM-ART : Growing Self Organizing Map for sequence clustering and classification, European Conference on Artificial Intelligence (ECAI 2004), Valencia, Spain, August 2004.

[3] F. Zehraoui and Y. Bennani, M-SOM: Matricial Self Organizing Map for Sequence clustering and classification, International Joint Conference on Neural Networks (IJCNN 2004), Budappest, Hungary, July 2004.

[4] T. Kohonen, Self-Organizing Maps, Springer Series in Information Sciences, Vol. 30, Springer, Berlin, Germany, 2001.

[5] G.A. Carpenter and S. Grossberg, The art of adaptive pattern recognition by a self-organizing neural network, IEEE Computer, 21(3), p. 77-88, March 1988.

[6] Z. Jrad., B. Benmammar, J. Correa, F. Krief , N. Mbarek. A user assistant for QoS negotiation in a dynamic environment using agent technology. Second IFIP International Conference on Wireless and Optical Communications Networks WOCN 2005. Dubai, United Arab Emirates UAE, March 2005.

[7] Z. JRAD. Intelligence artificielle et réseaux. Hermès Science. Traité IC2, Rédaction du chapitre "Les Plate-formes Multi-agents », Paris, 2004.

[8] Bellifemine F., Caire G., Poggi A., Rimassa G. JADE a white paper. Expvolume 3 n°3. Septembre 2003. http:// exp.telecomitalialab.com.

[9] Z. Jrad. And F. Krief. An Intelligent Interface for the Dynamic Negotiation of QoS in ARCADE. ACS/IEEE International Conference on Pervasive Services. ICPS'2004. Beirut, Liban. Juillet 2004.

[10] Z. Jrad. Spécification d'une interface intelligente pour la prochaine génération de réseaux IP. Journées Scientifiques DNAC 2005. Les réseaux IP : synthèse et perspective, contrôle, gestion, mobilité et sécurité. Beyrouth, Liban. Avril 2005.

1.4. Publications

[1] **Z. Jrad**, **F. Krief** and **B. Benmammar** "An Intelligent User Interface for the Dynamic Negotiation of QoS" 10th International Conference on Telecommunications, February 23 – March 1, 2003, Tahiti, Papeete, French Polynesia.

[3] **F. Krief** et **Z. Jrad**. An Intelligent Environment for Dynamic negotiation, provisioning and control of QoS in IP networks. Colloque Gestion de Réseau et de Service. GRES'2003. Fortaleza, Brésil. Février 2003.

[2] G. Klein and **F. Krief**. Mobile Agents for Dynamic SLA Negotiation. International Workshop on Mobile Agents for Telecommunication Applications. MATA'2003. Lecture Notes on Computer Science, Springer. Marrakech, Maroc. October 2003.

[4] **Z. Jrad** And **F. Krief**. An Intelligent Interface for the Dynamic Negotiation of QoS in ARCADE. International Conference on Pervasive Services. ICPS'2004. Beyrouth, Liban. Jully 2004.

[5] **Z. Jrad**, **B. Benmammar**, **J. Corréa**, **F. Krief** and **N. Mbarek**. "A User Assistant for QoS Negotiation in a Dynamic Environment Using Agent Technology". Second IEEE and IFIP International Conference on Wireless and Optical Communications Networks (WOCN 2005). March 6-8, 2005, Dubai, United Arab Emirates UAE.

[6] **Z. Jrad**, "Les réseaux sans fil et la nouvelle signalisation IP". École DNAC de printemps, Liban, du 11 au 18 décembre 2005.

[7] **Z. Jrad**. Spécification d'une interface intelligente pour la prochaine génération de réseaux IP. Journées Scientifiques DNAC 2005. Les réseaux IP : synthèse et perspective, contrôle, gestion, mobilité et sécurité. Beyrouth, Liban. Avril 2005.

[8] **Z. JRAD**. Intelligence artificielle et réseaux. Hermès Science. Traité IC2, Rédaction du chapitre "Les Plate-formes Multi-agents », Paris, 2005.

2. Utilisation de SLN NSLP pour l'offre de service

Dans le livrable 4.1, nous avons proposé un nouveau protocole de négociation que nous avons appelé SLN NSLP (Service Level Negotiation NSLP). Ce protocole s'appuie sur l'environnement NSIS.
Dans cette section, nous commençons par rappeler les paramètres de niveau de service qui peuvent être utilisés par SLN NSLP pour la négociation d'un SLS. Ensuite, nous spécifions les interactions de notre protocole avec les autres protocoles de l'environnement NSIS, c'est-à-dire GIMPS et QoS NSLP pour l'offre de ce niveau de service. Enfin, nous présentons un cas d'utilisation pour la visioconférence ou encore la téléphonie sur IP avec SIP.

2.1. Paramètres de SLS négociés

Dans ce qui suit, nous présentons les différents paramètres que peut contenir un SLS qui va être négocié grâce à notre protocole SLN NSLP. La présence d'un paramètre dans les messages SLN NSLP échangés entre les SNE est indiquée par un drapeau dans le deuxième champs de l'entête des messages SLN NSLP. La position d'un drapeau est spécifique à un attribut du SLS et peut être mise à 0 ou 1, auquel cas un objet TLV caractérisant ce paramètre DOIT être présent à la suite de l'entête du message SLN NSLP. La position des objets TLV qui définissent les paramètres du SLS négocié suivent le même ordre que les drapeaux de l'entête qui signalent leurs présence. Une fois négociés grâce aux échanges de messages SLN NSLP, les attributs du SLS vont être mappés dans les champs d'un QSPEC [1] définit par un QSM (QoS Signaling Model) spécifique à un modèle de QoS (Diffserv, Intserv...). Les paramètres de SLS que nous présentons sont fortement liés à l'aspect qualité de service mais SLN NSLP reste ouvert et assez générique pour échanger des paramètres de SLS autre que ceux en rapport avec la QoS et ce grâce à l'indépendance des objets qui définissent ces paramètres vis-à-vis du fonctionnement du protocole. Il suffit donc, suivant nos besoins, d'étendre les attributs du SLS à des aspects liés à la sécurité ou encore à la mobilité et définir les objets TLV correspondants qui vont être transportés dans les messages SLN NSLP. Néanmoins, ces paramètres ne vont pas forcement être mappés dans le QSPEC, qui est, à priori, conçu pour régler le problème de garantie de QoS, et il faudra donc penser aux interactions nécessaires pour garantir un niveau de sécurité ou de mobilité des utilisateurs. C'est pourquoi, nous nous limitons ici aux paramètres en relation avec la QoS.

2.1.1. Identification du trafic

Ce paramètre nous permet d'identifier le flux IP auquel on va garantir un certain niveau de service. Il peut s'agir d'un micro flux ou d'un agrégat de micro flux, dans les deux cas, c'est un ensemble de datagrammes IP qui a au moins une caractéristique en commun :

- L'adresse IP source et/ou destination : une seule adresse, un ensemble d'adresses ou un préfixe.
- Le numéro de port source et/ou destination.
- L'ID du protocole associé au flux IP tel que TCP ou UDP.
- Une valeur de DSCP qui permet d'identifier un agrégat de flux dans un domaine qui utilise DiffServ comme modèle de QoS.
- Un Label MPLS qui permet d'identifier un micro flux ou un agrégat de micro flux appartenant à un LSP d'un domaine qui utilise la technique de commutation de label MPLS.

2.1.2. Scope

Le Scope nous permet de préciser les bords géographiques entre lesquels le niveau de service va être négocié pour garantir une certaine qualité de service aux flux circulant entre un point d'entrée (Ingress) et un point de sortie (Egress). Ainsi on peut dire que Scope = (Ingress, Egress). Vue que SLN NSLP est un protocole de signalisation dans la bande, on définit pour le Scope un seul Ingress et un seul Egress (1,1) qui définissent le chemin que vont suivre les messages SLN NSLP pour la négociation du niveau de service End to Edge ou Edge to Edge. Ces deux points peuvent ou non appartenir à un même domaine, auquel cas on parle de négociation inter ou intra domaine.
On peut considérer l' exemple suivant où le SLS doit être négocié entre le domaine LIPN et celui du LIP6, on aura alors un scope spécifié par:
Ingress = 194.254.163.24
Egress = 132.227.60.254

2.1.3. Temps de service

Le temps de service, appelé aussi Service Schedule, nous informe sur la période au cours de laquelle un niveau de service doit être valable pour un flux identifié suivant un scope bien déterminé. Ce temps de service peut être immédiat (Exemple : téléphonie), auquel cas le temps de début est nul. Il peut être périodique et il est spécifié alors par l'heure du jour, le jour de la semaine, le mois de l'année. Enfin on peut, grâce à un temps de service différé, faire une réservation à l'avance des ressource afin de garantir leur disponibilité le moment voulu (Exemple : Vidéo conférence) : cette réservation à l'avance n'empêche pas les ressources d'être utilisées avant le début du temps de service.
On peut présenter le temps de service comme :

<Temps de service> *= <Type> <Mois_D> <Jour_D> <Temps_D> <Mois_F> <Jour_F> <Temps_F>*

Type : Indique si le temps de service correspond à une réservation immédiate, périodique ou à l'avance.
D : Le début du temps de service.
F : La fin du temps de service.

2.1.4. Garantie de Performance

Ce paramètre contient les attributs les plus utilisés pour spécifier les caractéristiques QoS d'un flux IP. Ils sont au nombre de cinq : Délai, Gigue (Jitter), Taux de perte (Packet Loss Ratio), Débit (Throughput) et peuvent être quantitatifs avec des valeurs numériques données à ces attributs ou qualitatifs, auquel cas des adjectifs variant entre excellent et mauvais sont utilisés pour spécifier ces attributs.

<Garantie de Performance> *= <Type> <Délai> <Gigue> <Taux De Perte> <Débit>*

Avec :
Type : Indique si les valeurs sont qualitatives ou quantitatives.
Délai : Le délai d'acheminement d'un paquet IP entre l'Ingress et l'Egress du scope.
Gigue : La variation du délai d'acheminement d'un paquet IP entre l'Ingress et l'Egress.
Taux de Perte : Le pourcentage de paquets IP qui sont perdus entre l'Ingress et l'Egress.
Dédit : Débit relatif aux paquets appartenants à la couche IP mesuré entre l'Ingress et L'Egress spécifiés par le scope.

Dans le cas de valeurs quantitatives : le délai et la gigue s'expriment en ms, le taux de perte est un pourcentage et le débit en bit/s. Dans le cas de valeurs qualitatives, des adjectifs comme excellent, bon, moyen et mauvais peuvent être utilisés.

2.1.5. Description et conformité du trafic

Ce paramètre comporte un ensemble d'indicateurs qui décrivent les caractéristiques de conformité que doivent satisfaire les paquets d'un flux IP afin qu'on leur fournisse le niveau de service spécifié dans le paramètre Garantie de Performance. On parle alors d'un trafic « in-profile » ou « out-of-profile ». Ce test de conformité peut être réalisé par le biais d'un algorithme de conformité tel que le Token Bucket.

> ***<Description et conformité du trafic>*** *= <Token Bucket Rate(r)> <Token Bucket Size(b)> < Peak Data Rate(p)> <Minimum Policed Unit(m)> <Maximum Packet Size (M)>*

r : Débit des jetons du Bucket
b : Taille de la file d'attente du Bucket en octet pour un débit r
p : Le débit crête du trafic en bit/s
m : La plus petite taille de paquet IP acceptée (en octet)
M : La plus grande taille de paquet IP acceptée (en octet)

2.1.6. Traitement d'excès

Dans le cas où un trafic présente des paquets non conformes aux attributs spécifiés dans le paramètre description et conformité du trafic, ces paquets sont considérés hors profil et sont exposé au traitement d'excès spécifié dans ce paramètres de SLS. Un traitement d'excès peut consister en l'élimination des paquets hors profil (Dropping), ou encore retarder les paquets hors profil et les transmettre au bon moment afin de rendre le flux de données de nouveau conforme (In Profile) : c'est ce qu'on appelle le lissage (Shaping). Enfin une dernière alternative consiste à dégrader le niveau de service des paquets non conformes (Out of Profile) en les affectant à des classes de service inférieures à celle négociées par le SLS et ce grâce, par exemple, à un remarquage (Remarking) des paquets hors profil.

> ***<Traitement d'excès>*** *= <Dropping> <Shaping> <Remarking>*

Avec:
<Dropping> = Elimination
<Shaping> = Lissage
<Remarking> = Remarquage

2.1.7. Mode de négociation et Intervalle de renégociation

Pour la négociation de SLS avec SLN NSLP, nous proposons deux modes :

- SLS Prédéfinis : La négociation va porter sur un ensemble de SLS avec des valeurs fixées ou plages de valeurs pour certains paramètres que peuvent supporter les SNE présents sur le chemin des données.
- SLS non prédéfinis : Les valeurs que peuvent prendre les paramètres sur lesquels va porter la négociation peuvent ou non avoir des contraintes à satisfaire.

Le mode de négociation est indiqué par deux drapeaux dans la deuxième partie de l'entête des messages SLN NSLP avec les valeurs suivantes :
00 : Mode Non Prédéfini sans contraintes
01 : Mode Non Prédéfini avec contraintes
11 : Mode Prédéfini

L'intervalle de renégociation est un paramètre utilisé dans les messages SLN NSLP pour indiquer le temps au bout duquel un SLS négocié peut être renégocié. Il dépend fortement des acteurs mis en jeu lors de la procédure de négociation, en effet un SLS négocié entre deux fournisseurs demande un temps considérable avant que les éléments du réseau ne soient configurés pour garantir un niveau de service, vu l'importance du trafic généré par un fournisseur. Au contraire un particulier génère beaucoup moins de trafic et, par suite, son SLS peut être renégocié dans un temps beaucoup plus court. Une valeur plus ou moins importante de l'intervalle de renégociation entraîne une dynamicité plus ou moins importante du SLS. Cet attribut peut être transporté dans un objet TLV qui comporte entre autres les indicateurs suivants.

<Intervalle de Renégociation> = *<Heures> <Minutes>*

Avec :

Heures : Le nombre d'heures avant qu'un SLS ne soit renégocié.

Minutes : Le nombre de minutes avant qu'un SLS ne soit renégocié.

2.1.8. Priorité et Fiabilité

La priorité négociée pour un flux de données permet aux paquets de ce flux d'être traités plus ou moins rapidement suivant son importance. Dans la deuxième version du draft portant sur la spécification du Qspec, on se propose de déterminer une forme générique pour la priorité [2] :

<Priority> = *<Reservation Priority> <Setup Priority> <Holding Priority>*

La fiabilité nous permet d'avoir une garantie supplémentaire sur la limitation dans le temps de l'effet d'une panne dans le réseau sur l'offre de niveau de service spécifiée dans le paramètre Garantie de Performance.

<Fiabilité> = <Mean Down Time (MDT)> <Mean Time To Repair(MTTR)>

Avec:

MDT : Le maximum du temps moyen de panne par an (minutes)

MTTR : Le maximum du temps moyen pour le rétablissement d'un niveau de service (secondes)

2.2. Interactions de SLN NSLP avec GIMPS et QoS NSLP

SLN NSLP est en relation verticale avec la couche NTLP et en relation horizontale avec une application de signalisation de même niveau, à savoir QoS NSLP. D'une part, pour le transport et le routage de ses messages de négociation, SLN NSLP a besoin des services de la couche NTLP afin de lui assurer le bon acheminement des informations de négociation vers les noeuds qui implémentent SLN NSLP et qui se trouvent sur le chemin des données. Ceci est rendu possible grâce à la définition d'une API entre GIMPS et l'application de signalisation, en particulier SLN NSLP pour notre cas.

D'autre part, une fois la négociation achevée avec succès, SLN NSLP doit compter sur une autre application de signalisation, à savoir QoS NSLP, pour la réservation des ressources conformément au SLS qui a été accepté lors de la procédure de négociation. Ceci est rendu possible grâce à l'utilisation de l'objet QSPEC, opaque à QoS NSLP, dans lequel on va mapper les paramètres de SLS, conformément au modèle de QoS adopté tout au long du chemin des données.

Dans ce qui suit, nous détaillons la collaboration entre SLN NSLP, GIMPS et QoS NSLP en décrivant davantage l'utilisation de cet API ainsi que la structure de l'objet QSPEC.

2.2.1. API entre SLN NSLP et GIMPS

Différentes primitives composent l'API qui sert de lien entre GIMPS et la couche NSLP. SLN

NSLP est une signalisation pour la négociation de niveau de service qui appartient à la couche NSLP et, par la suite, elle utilise cette API pour communiquer avec la couche basse de l'architecture NSIS, à savoir NTLP. Nous allons, maintenant, étudier comment les paramètres des primitives de l'API permettent de satisfaire les besoins de bon fonctionnement de SLN NSLP.
Les paramètres NSLP-Data et NSLP-Data-Size dans les primitives SendMessage et RecMessage permettent à SLN NSLP d'envoyer et recevoir des messages de négociation. Ces messages sont pris en charge par la couche NTLP afin d'assurer leur routage et transport grâce aux paramètres MRI et Direction tout en respectant les besoins de fiabilité et de sécurité définis par SLN NSLP par le biais du paramètre Transfer-Attributes.
L'identification de la session pour laquelle SLN NSLP génère une signalisation est réalisée par le paramètre Session-ID présent dans les messages SendMessage et RecMessage. Cet attribut est généralement fixé par SLN NSLP et suffisamment aléatoire pour ne pas être confondu avec un autre identificateur de session.
L'argument SII-Handle présent dans les messages SendMessage et RecMessage permet d'identifier un changement dans l'adresse du noeud adjacent et, par la suite, un changement de route.
Dans le cas d'un changement de route SLN NSLP peut être informé grâce à une valeur particulière de l'attribut Network-Notification-Type de la primitive Network Notification.
SLN NSLP peut se rendre compte qu'elle est la dernière sur le chemin des données grâce à la primitive MessageDeliveryError en spécifiant une valeur particulière pour l'argument Error-Type.
Ainsi ces primitives offrent un ensemble d'arguments qui facilitent l'échange d'informations entre SLN NSLP et GIMPS, permettant ainsi le bon acheminement des messages SLN NSLP entre les différents SNE présents sur le chemin des données.

2.2.2. Interaction entre SLN NSLP et QoS NSLP

Il s'agit, ici, d'une collaboration horizontale entre ses deux applications de signalisation appartenant à la même couche NSLP définie dans l'architecture NSIS. Une procédure de négociation réussie par SLN NSLP doit être suivie par une réservation de ressources par QoS NSLP pour garantir une disponibilité, que ce soit immédiate ou différée suivant le temps de service, des ressources sur le chemin des données. Sachant que lors de la négociation avec SLN NSLP, une interaction a eu lieu avec les différents modules du RMF, à savoir la composante Admission Control pour s'assurer de l'existence des ressources et le Policy Control pour les autorisations nécessaires, QoS NSLP n'a plus à réaliser ces contrôles et se contente de présenter l'identificateur du SLS (SLS ID) au RMF pour se voir accepté la réservation des ressources correspondant au niveau de service négocié au préalable. Ce niveau de service négocié grâce à SLN NSLP est spécifié par les différents attributs du SLS, cependant ses paramètres sont loin d'être compréhensibles par QoS NSLP et dépendent fortement du modèle de QoS adopté par le domaine que traverse un message QoS NSLP. Une solution à ce problème est l'utilisation de l'objet QSPEC, comme intermédiaire entre SLN NSLP et QoS NSLP, en effet les paramètres de SLS sont mappés dans le QSPEC approprié au modèle de QoS et opaque à QoS NSLP. Les objets de ce QSPEC et la façon avec laquelle il est traité dans les noeuds QNE sont spécifiés par le QSM (QoS Signaling Model) qui décrit la façon avec laquelle on utilise QoS NSLP et QSPEC pour la réservation de ressources dans un domaine adoptant un modèle de QoS déterminé (Diffserv, Intserv...). Si le chemin de données passe par plusieurs domaines employant des modèles de QoS différents, les QSPEC peuvent être empilés à l'entrée d'un domaine et enlevés à la sortie. Seul le QSPEC se trouvant au sommet de la pile est envoyé par le QNE au RMF. Dans le cas où le QSPEC n'est pas compréhensible ou ne peut être traité par le RMF, le QNE envoie un message Response indiquant cette erreur et ne doit

pas acheminer le message qui contient ce QSPEC vers le noeud adjacent.
Comme tout objet NSIS le QSPEC a une structure en TLV avec les champs suivants :

Type : un objet QSPEC (16 bits)
Longueur : variable (16 bits)
Valeur = <QSM ID (32 bits)> <QSPEC (variable)>

La deuxième partie du champ Valeur contient des paramètres qui dépendent du QSM. Nous allons maintenant décrire un modèle de QSPEC [1] qui est assez générique pour être retenu comme base pour les QSPEC spécifiques à un modèle de QoS.
Le Template de QSPEC que nous décrivons contient des paramètres qui peuvent changer d'un noeud à un autre (mutable) et d'autres qui ne le sont pas (immutable).

<QSPEC> = <QSM Control Information> <QoS Description>

Les informations de contrôle permettent d'avoir des fonctions de signalisation autres que celles définies dans QoS NSLP.

<QSM Control Information> = <Hop Count> <Service Schedule>

Le paramètre Temps de Service négocié par SLN NSLP peut être mappé dans le Service Schedule pour une réservation à l'avance ou immédiate des ressources. Le Hop Count nous permet de limiter le Scope du QSPEC à un nombre limité de QNE.

<QoS Description> = <QoS Desired> <QoS Available> <QoS Reserved>
<Minimum QoS>

Le sous objet qui nous intéresse le plus est QoS qu'on veut avoir après la négociation du niveau de service avec SLN NSLP.

<QoS Desired> = <BW> <Token bucket> <QoS Class> <Priority>

D'autres paramètres comme <Transfer Delay> <Delay Variation> <Packet Loss Ratio> sont en train d'être discutés pour faire partie de ce modèle de QSPEC. Ainsi les paramètres Garantie de performance et Description et conformité du trafic sont facilement traduits conformément à QoS Description de l'objet QSPEC.
Grâce à l'ensemble de ses paramètres génériques et d'autres optionnels, car spécifiques à un QSM, nous pouvons mapper les paramètres de SLS négociés par SLN NSLP dans l'objet QSPEC afin de garantir un certain niveau de service.

2.3. Cas d'utilisation de SLN NSLP avec SIP

Dans un environnement Internet où nous pouvons communiquer par le biais de plusieurs médias (Texte, vidéo, Audio...), SIP nous permet de créer, modifier et terminer des sessions avec un ou plusieurs participants et ce pour des applications comme la téléphonie sur IP, la diffusion de contenu multimédia ou encore la visioconférence.
Nous présentons dans cette partie deux cas d'utilisation de SLN NSLP avec SIP. Dans le premier cas, SLN NSLP négocie à l'avance le niveau de service requis pour une visioconférence avec SIP qui sera programmée dans le futur suivant un Temps de Service. Dans le deuxième cas, nous négocions le niveau de service pour une utilisation immédiate des ressources pour la téléphonie sur IP avec SIP.

2.3.1. Visioconférence

Le premier cas d'utilisation de SLN SLP concerne la négociation de niveau de service pour une visioconférence qui va permettre à un chercheur se trouvant dans un laboratoire à Paris de participer à un séminaire qui se déroule à Bordeaux. Pour se faire, il ne va pas réserver un billet de TGV mais négocier un niveau de service grâce à SLN NSLP et réserver avec QoS NSLP les ressources correspondant au SLS en mappant les paramètres de ce dernier dans l'objet QSPEC. La négociation va alors se faire en avance par rapport à la date à laquelle la visioconférence aura lieu, ceci est spécifié dans le paramètre de SLS Temps de Service. Nous voulons que cette

visioconférence qui sera initiée par SIP dispose d'un bon niveau de service qui correspondrait à un MOS (Mean Opinion Score) supérieur à 4 sur une échelle de 1 à 5, c'est ce qui nous amène à demander des limites pour les attributs du paramètre Garantie de performance. En effet le délai doit être inférieur à 150 ms, la gigue inférieur à 30 ms et le taux de perte inférieur à 1%. Le débit, quant à lui, dépend essentiellement des codecs utilisés. Les flux audio vont être codés suivant la recommandation ITU-T G722 et transportés par le protocole RTP (Real Time Protocol) grâce à une adaptation défini par l'IETF dans [3]. Les flux vidéo vont avoir une résolution de 30 images par seconde avec une taille de l'image suivant le format CIF (Common Intermediate Format : 352x288) en utilisant le codec H261 dont le flux de sortie va être transporté par une adaptation de RTP [4]. Le codec vidéo H261 avec la résolution que nous avons choisi demande un débit de 640 kbit/s alors que le codec audio G722 demande un débit de 56 kbit/s. Ainsi le SNI va négocier grâce au message Negotiate de SLN NSLP un SLS qui satisfait les besoins que nous avons définis. (Voir tableau 1)

Identification du trafic	Scope	Temps de Service	Garantie de Performance	Mode / intervalle
@IP Host A @IP Host B	@IP SNI @IP SNR	9h→17h Lundi→Jeudi	D<150ms ; G<30ms ; P<1% ; D= 696 Kbit/s	00 / 30 ms

Tableau 1 : SLS demandé

A la réception du message Negotiate, le SNR va fixer les valeurs de SLS qu'il peut garantir et les envoyer au SNI via un message Revision en précisant un identificateur pour cet SLS. (Voir tableau 2)

Identification du trafic	Scope	Temps de Service	Garantie de Performance	Mode / intervalle
@IP Host A @IP Host B	@IP SNI @IP SNR	9h→17h Lundi→Jeudi	D=100ms ; G=20ms ; P=0,1% ; D= 704 Kbit/s	00 / 30 ms

Tableau 2 : SLS accepté

Cette proposition est acquittée par le SNI par un message Response et le SLS ID est enregistré. Suite à cette négociation avec SLN NSLP, vient maintenant la procédure de réservation avec QoS NSLP. Les paramètres de SLS sont mappés dans un QSPEC conforme au QSM du domaine et la réservation se fait sur le chemin des données grâce au message Reserve et ce dans les deux sens puisque nous voulons réserver pour un flux bidirectionnel. Cette réservation est effectuée à l'avance puisque la visioconférence aura lieu à une date ultérieure ce qui n'empêche pas les ressources d'être utilisées par d'autres flux avant le début du service. A la date de la visioconférence, la communication va être établie grâce à un échange classique avec les requêtes de SIP avec la garantie que les ressources nécessaires seront disponibles pour permettre au chercheur parisien de bien suivre et participer à la conférence.

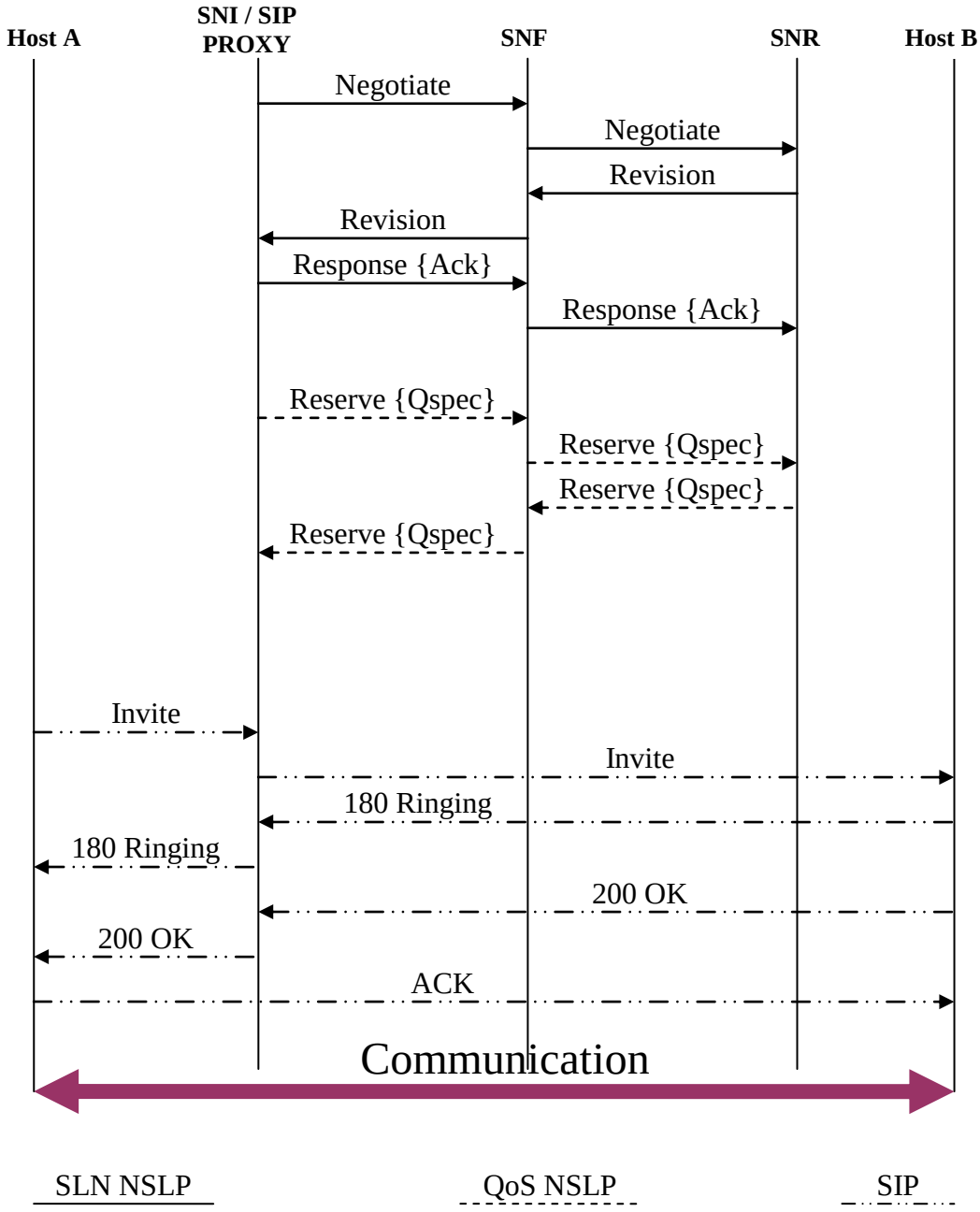


Figure 1 : Négociation pour une visioconférence

2.3.2. Téléphonie sur IP

Nous proposons dans ce qui suit un deuxième cas d'utilisation de la négociation avec SLN NSLP. Il s'agit d'une négociation qui va être initiée, suite à la demande d'établissement d'une session multimédia avec SIP de la part de l'utilisateur.

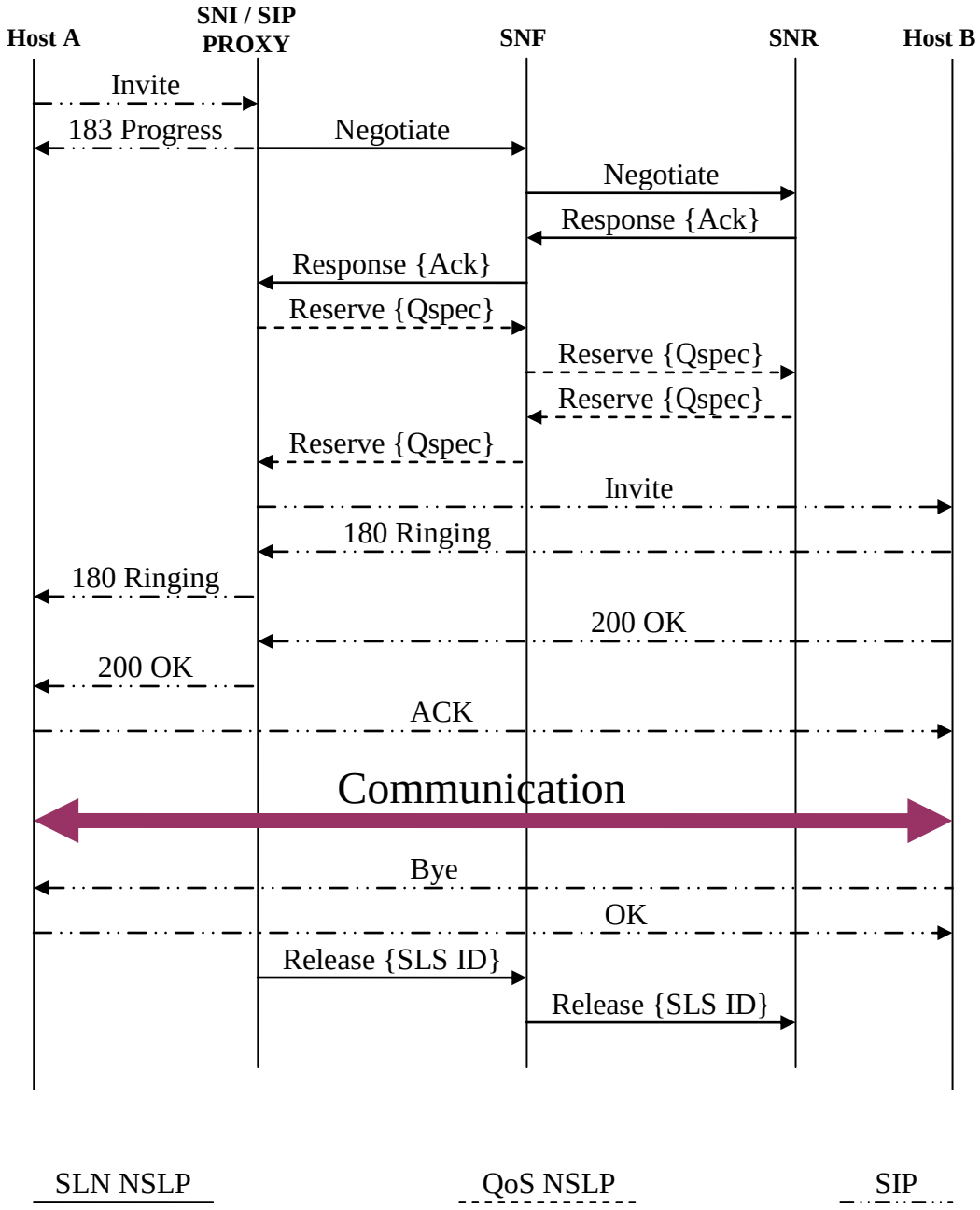


Figure 2 : Scénario 1 pour la téléphonie

Nous choisissons pour notre exemple l'application de téléphonie sur IP qui va permettre à un utilisateur A, initiateur de l'appel, de joindre un utilisateur B. Les contraintes en terme de délai, gigue et perte sont identiques à ceux défini pour la visioconférence. Le codeur qui va être utilisé est le G722 et, par suite, un débit de 56kbit/s sera suffisant. Le Temps de Service sera immédiat, vu que l'utilisation des ressources aura lieu juste après la procédure de négociation et de réservation. Nous avons défini deux scénarios : Dans le premier, la négociation et la réservation se font avant de s'assurer de la présence d'une composante SIP au niveau de l'utilisateur B. En effet dés la réception du message Invite par le Proxy SIP, qui est au même temps le SNI, ce message est bloqué et un message 183 Progress est envoyé à A pour lui dire que nous sommes en train de traiter sa requête. Au même temps, le SNI initie la procédure de

négociation pour voir s'il peut garantir la disponibilité immédiate des ressources requises par le SLS. Une fois un accord établi les ressources sont réservées par QoS NSLP et le SIP Proxy envoie un message Invite à B. Le téléphone sonne alors et un acquittement est envoyé pour permettre le début de la conversation. A la fin de la communication B envoie un message Bye à A qui répond par un OK. Le SLS est alors résilié par le SNI par un message Release émis vers le SNR. Ce scénario est adapté au cas où nous ne sommes pas sûr de la disponibilité des ressources mais sûr de l'existence d'une composante SIP chez B. L'inconvénient est que nous sommes obligés de résilier un SLS que nous n'avons pas utilisé dans le cas où B n'est pas présent. Nous avons alors pensé à un autre scénario d'échange, en effet la procédure de négociation n'est lancée que lorsque B nous répond par un message 200 OK, on peut bloquer ce message dans le SIP Proxy et négocier avec SLN NSLP et réserver avec QoS NSLP. Une fois ces deux procédures effectuées, nous pouvons commencer la communication. L'inconvénient réside dans le fait que la procédure de négociation peut échouer et, par la suite, nous devons nous contenter d'une qualité Best Effort ou encore ne pas établir la communication alors que notre vis-à-vis a été déjà alerté. L'intérêt de notre protocole de négociation est qu'à tout moment on peut demander à renégocier et par conséquent obtenir une QoS qui pouvait par exemple faire défaut au début de la communication.

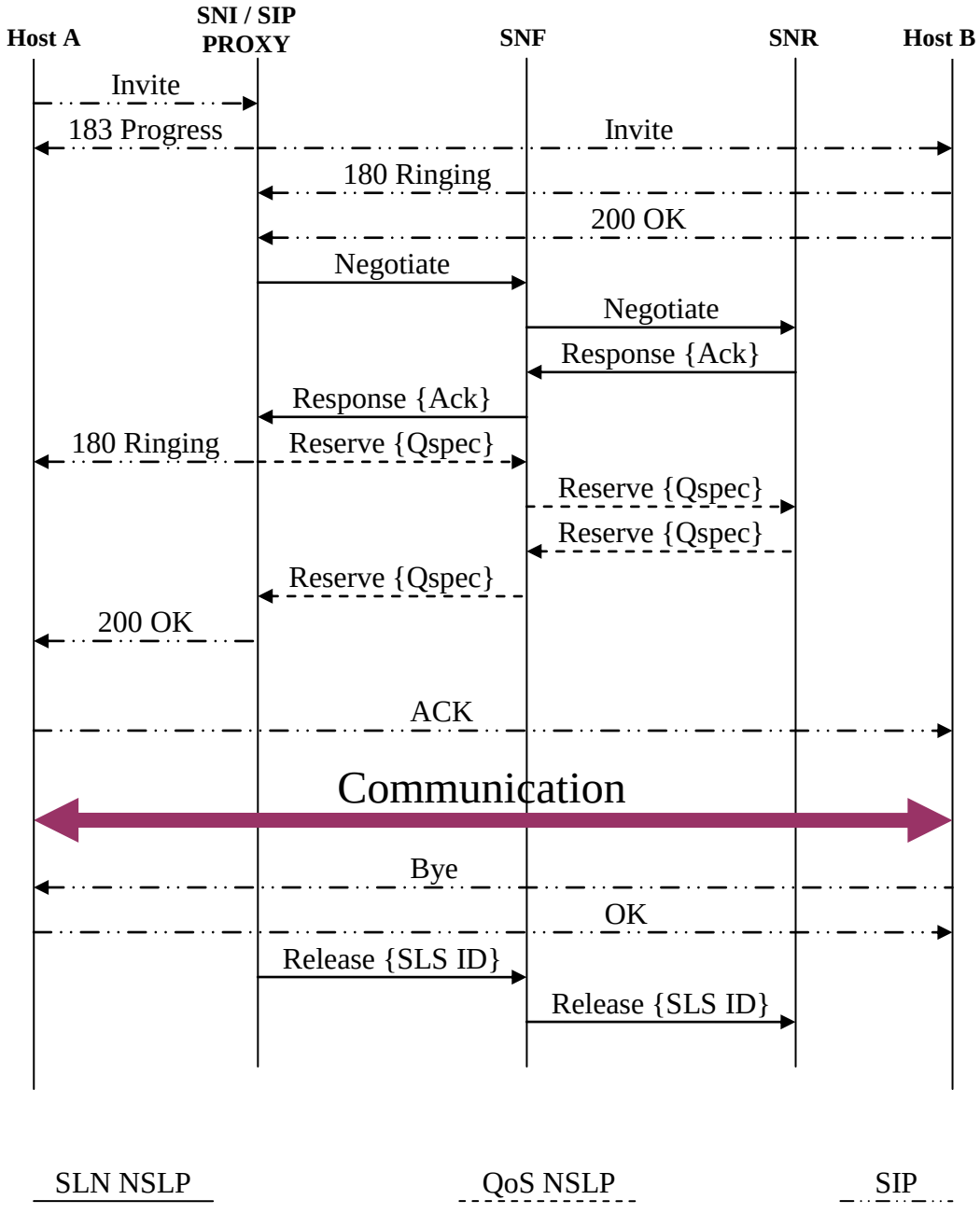


Figure 3 : Scénario 2 pour la téléphonie

2.4. Références

[1] J. Ash, A. Bader and al, « QoS NSLP QSpec Template », draft-ietf-nsis-qspec-01, Juillet 2004

[2] D. Awduche, L. Berger and al, «RSVP-TE: Extensions to RSVP for LSP Tunnels», RFC 3209, Décembre 2001.

[3] P. Luthi, « RTP Payload Format for ITU-T Recommendation G.722.1», RFC 3047, Janvier 2001.

[4] T. Turletti, C. Huitema, « RTP Payload Format for H.261 Video Streams», RFC 2032,

Octobre 1996.

2.5. Publications

[1] **N. Mbarek**, **F. Krief**, « Négociation du niveau de service dans un environnement NSIS : SLN NSLP», 18eme Congres DNAC, Novembre 2004, Paris, France
[2] **N. Mbarek**, **F. Krief**, « SLN NSLP:A Service Level Negotiation Signaling Protocol in the NSIS Environment», 12th IEEE International Conference on Telecommunications (ICT'2005), 3-6 May 2005, Capetown, South Africa.

II. Impact de la mobilité du terminal sur la signalisation

Dans le livrable 4.1, nous avons étudié l'impact de la mobilité du terminal sur la signalisation NSIS. Pour minimiser la dégradation de services durant le handover, nous avons proposé d'utiliser les messages de QoS NSLP afin de faire des réservations à l'avance, cette réservation est faite selon un objet MSpec, qui est inclus dans un profil de mobilité.
Cette partie du livrable détaille notre approche pour améliorer la QoS dans un réseau sans fil. Elle décrit également le profil de mobilité de l'utilisateur qui contient le Mspec et présente un système hybride pour la détermination des futures localisations du terminal mobile. De plus, le protocole MQoS NSLP et la procédure de handover sont détaillés dans le cas de deux terminaux mobiles et un scénario de handover vertical est proposé dans le cadre des futurs réseaux de $4^{ème}$ génération.
Avec notre approche, une fois le profil de mobilité de l'utilisateur déterminée, une réservation de ressources à l'avance est faite uniquement dans les localisations où le terminal mobile a une forte probabilité de se rendre. La détermination de cet ensemble de localisations est faite après une phase d'observation. Durant cette phase, le système ne peut pas réserver les ressources pour l'utilisateur car celui-ci est nouveau et donc son profil de mobilité est inconnu pour le système. Dans ce cas, nous proposons de faire appel à la technologie Agent afin d'améliorer la qualité de service pour cet utilisateur.

Le diagramme suivant représente le traitement associé à chaque cas :

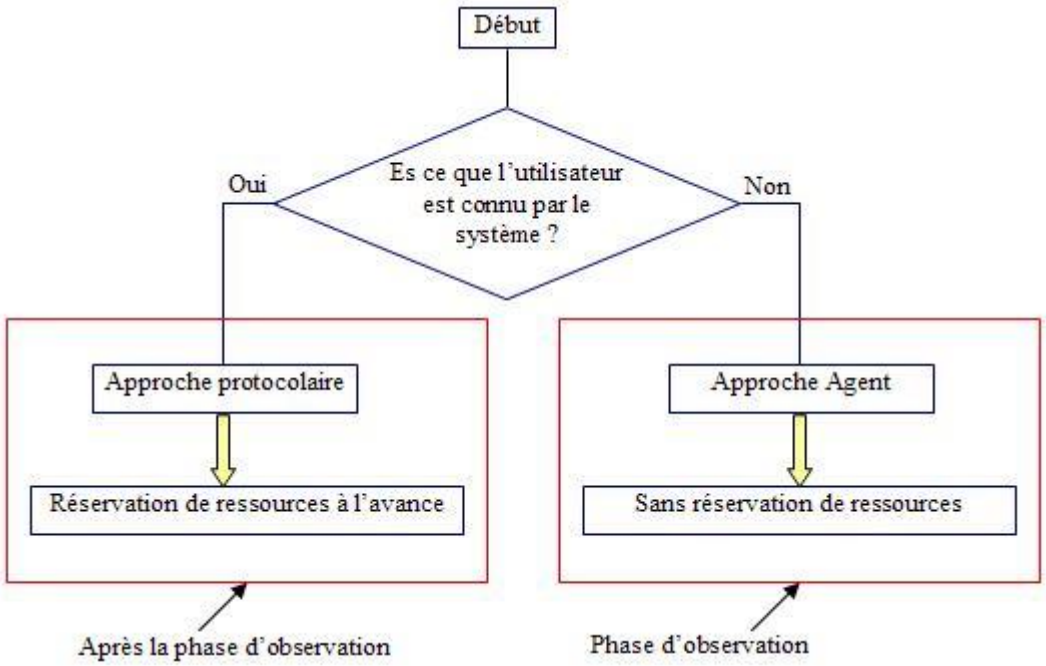


Figure 1. Les deux approches de la gestion de la QoS selon le type d'utilisateur

1. L'approche protocolaire

1.1. Le profil de mobilité

Cette section décrit notre proposition pour le profil de mobilité de l'utilisateur, ce profil est basé sur l'analyse du comportement de l'utilisateur afin de déterminer ses futures localisations. L'espace de mobilité de l'utilisateur est constitué de **N** cellules.

Le profil de mobilité de l'utilisateur est construit sur la base de son comportement/mouvement suite à **m** associations avec le système (avec les **N** cellules).

Pour la modélisation du système, nous avons opté pour les chaînes de Markov en temps continu. Les raisons de ce choix sont basées sur le fait que les chaînes de Markov sont des systèmes sans mémoire, le passage d'un état E_i à un autre état E_j ou *transition*, ne dépendant que de ces deux états et s'effectuant selon la probabilité conditionnelle : Prob (E_j / Ei) = P_{ij} (Probabilité de se trouver dans l'état E_j en fin de transition sachant qu'au début de la transition on était dans l'état initial E_i). Lors d'un handover, le passage d'une cellule à une autre ne dépendant que de ces deux cellules, cela correspond bien au comportement d'une chaîne de Markov.

Notre système est un modèle pouvant évoluer entre N états définis par l'ensemble:

C = (C_1, C_2, ...C_i.....C_n) qui représente l'ensemble des **N** cellules.

Le système est à l'état i = le terminal mobile se trouve dans la cellule C_i.

P_{ij} : la probabilité de transition de la cellule C_i vers la cellule C_j.

P_i (t_r) : la probabilité pour que le terminal mobile se trouve dans la cellule C_i à l'instant t_r.

Le but de ce profil est de construire un modèle comportemental pour l'utilisateur, il contient les informations suivantes :

- *Un identificateur unique de l'utilisateur (User_id*)

À travers cet identificateur, le système peut identifier l'utilisateur, il est soit connu pour le système ou bien nouveau.

- *Les préférences de l'utilisateur : User_P*

Cet attribut représente l'ensemble des préférences de l'utilisateur.

Exemple : quand l'utilisateur se déplace vers la cellule1, qui couvre un espace de détente, il commence toujours par le lancement d'un jeu vidéo durant 1h.

Les préférences de l'utilisateur sont déterminées après la *phase d'observation* durant laquelle le système observe le comportement de l'utilisateur.

Le diagramme suivant représente la détermination du User_P :

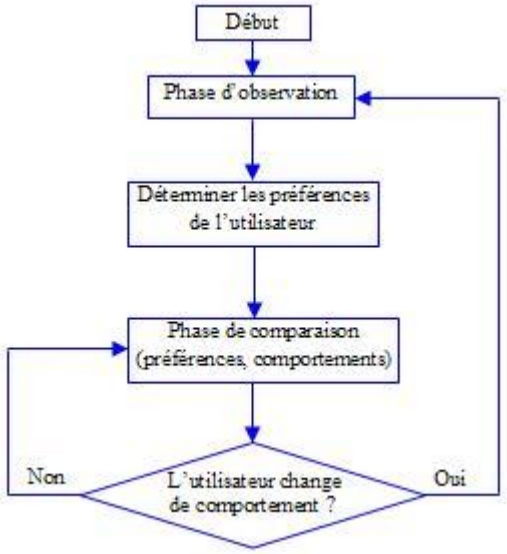


Figure 2. Les préférences de l'utilisateur

Le format proposé pour le User_P est le suivant :

User_P = < Preference ID> <Duration_P> <Cell_P> < QoS_level>

- *<Preference ID>*: un identificateur unique pour chaque préférence (le système peut détecter plusieurs préférences pour un utilisateur).
- *<Duration_P>: <start_P> <end_P>:* il détermine la période de temps durant laquelle la préférence de l'utilisateur est satisfaite.
- *<Cell_P>*: il détermine la cellule dont laquelle la préférence de l'utilisateur est satisfaite.
- *< QoS_level>:* c'est le niveau de QoS demandé par l'utilisateur pour chaque préférence.

- *$M = [P_{ij}]$ [N*N]*

C'est la matrice de transition qui contient les P_{ij}. Avant les **m** associations, les P_{ij} sont aléatoires.

On note **t** [i, j] : le nombre de transitions de la cellule i à la cellule j pendant les **m** associations avec le système.

g (i): le nombre de transitions qui ont comme point de départ la cellule i pendant les **m** associations. On le calcule de la manière suivante : $g(i) = \sum_{j=1}^{n} t[i, j]$.

Après les **m** associations, la probabilité de transition de la cellule i vers la cellule j est calculée de la manière suivante : $P_{ij} = t[i, j] / g(i)$.

- *$V = [P_i(t_o)]$ [N]*

C'est le vecteur qui contient les $P_i(t_o)$ (la probabilité pour que le terminal mobile se trouve dans la cellule C_i à l'instant t_o).

Avant les **m** associations, les $P_i(t_o)$ sont aléatoires. Après chaque association avec le système le temps est initié à t_o.

k (i) : c'est le nombre de fois que l'utilisateur se connecte dans la cellule i à l'instant t_o.

On a $\sum_{i=1}^{n} k(i) = m$ et $P_i(t_o) = k(i) / m$.

Le diagramme suivant représente la détermination de la matrice M et du vecteur V :

On suppose que le terminal possède un mécanisme pour se localiser et qu'il connaît à tout instant la cellule avec laquelle il s'associe.

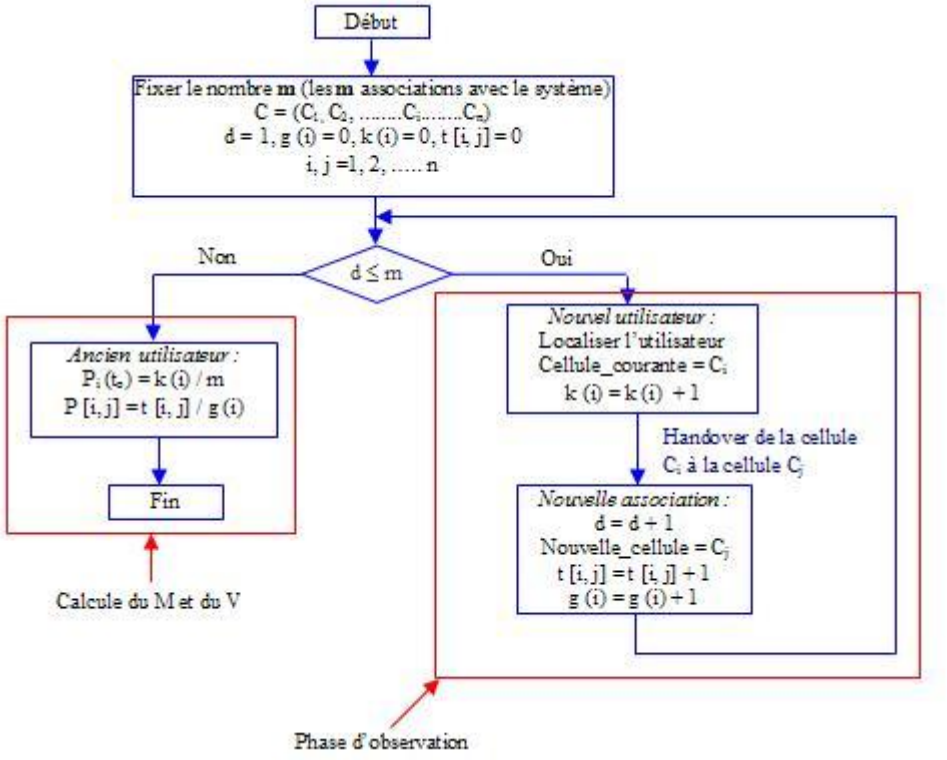


Figure 3. La détermination de la matrice M et du vecteur V

- *Le MSpec (Mobility Specification):*

Le MSpec représente la liste des localisations géographiques que le mobile va visiter pendant la durée de vie d'un flux. Il détermine les futures localisations du terminal mobile, le MSpec est un ensemble dynamique, il est modifié après chaque handover. Le format proposé pour le MSpec est le suivant : MSpec = <MSpec ID> <Duration> <Cell ID>. (Cet attribut sera inclus dans les messages de QoS NSLP).

- *MSpec ID* : un identificateur unique du MSpec.
- *Duration* : <start time>, <end time> : c'est l'intervalle de temps pendant lequel il est possible de déterminer les futures localisations du terminal mobile.
- *Cell ID* : <cell ID1>, <cell ID2>, <cell ID3>,....., <cell IDn>: c'est un ensemble d'identificateurs de cellules. On suppose que chaque cellule est identifiée par un identificateur unique.

P_{ij} est la probabilité de transition de la cellule C_i vers la cellule C_j.

P_i (t_r) est la probabilité pour que le terminal mobile se trouve dans la cellule C_i à l'instant t_r, on a $\sum_{i=1}^{n} P_i(t_r) = 1$.

P_j (t_{r+1}) : la probabilité pour que le terminal mobile se trouve dans la cellule C_j à l'instant t_{r+1}. On peut calculer cette probabilité à l'aide de la formule suivante :

$$P_j(t_{r+1}) = \sum_{i=1}^{n} P_i(t_r) * P_{ij}.$$

On définit θ ($0 \leq \theta \leq 1$) : un seuil fixe ou variable, permettant de sélectionner les cellules de plus grandes probabilités.

Le MSpec est défini comme suit : MSpec (t_r) = {C_j / P_j (t_{r+1}) ≥ θ }

Après le handover, on lance à nouveau le processus Markovien pour déterminer le nouveau MSpec.

Avant les **m** associations, le système ne calcule pas le MSpec parce que l'utilisateur est nouveau est le système ne possède pas les deux informations permettant de calculer le Mspec, à savoir la matrice M et le vecteur V.

Le diagramme suivant représente la détermination du MSpec à l'aide du profil de mobilité (après la phase d'observation).

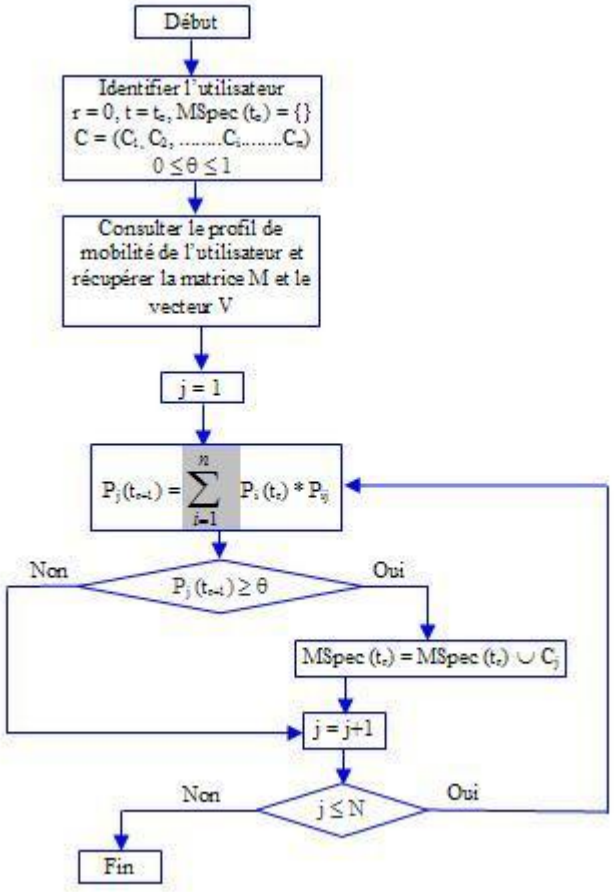


Figure 4. Prédiction du MSpec à l'aide du profil de mobilité

- *Variante pour la détermination du MSpec : un système hybride*

Les auteurs dans [1] proposent une nouvelle architecture pour la prédiction avec précision de la trajectoire d'un utilisateur mobile, ils ont nommé cette architecture Mobility Prediction Agent (MPA). La nouveauté apportée dans ce travail est d'une part l'incorporation d'informations cruciales comme des vraies cartes géographiques et d'autre part, l'utilisation de théorie mathématique comme moyen de raisonnement. Les cartes géographiques utilisées sont les SCM (Spacial Conceptual Map). Cette combinaison de données et de raisonnement vise à améliorer la capacité globale de l'algorithme de prédiction et à prendre en considération l'incertitude et les changements éventuels. L'architecture globale de ce modèle de prédiction est représentée sur la figure 5 :

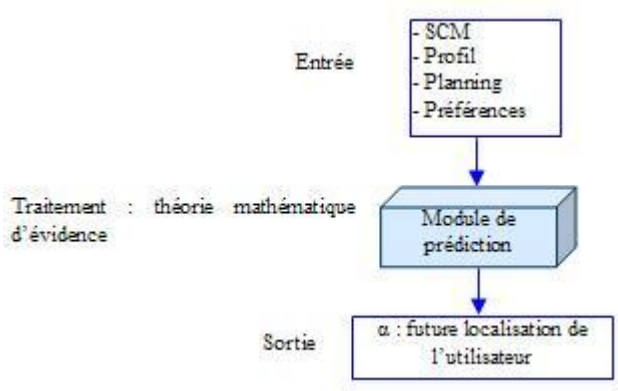


Figure 5. Définition fonctionnelle du module intelligent de prédiction

A la sortie du module de prédiction, la future localisation de l'utilisateur est représentée par une variable α qui s'appelle WEA (Way Elementary Area). Le module de prédiction est capable de déterminer le chemin entre $\alpha_{courant}$ et $\alpha_{prédit}$ avec tous les WEA du parcourt.

Les chaînes de Markov identifient le MSpec comme un ensemble de cellules, selon la valeur de θ. Une variante de cette méthode consiste à sélectionner une seule cellule qui a la plus grande probabilité. Formellement, on sélectionne la cellule C_k correspond à la probabilité P_k qui vérifie la contrainte : $\mathbf{P_k\,(t_{r+1}) = Max\ P_j\,(t_{r+1})\ /\ j = 1, 2, \ldots\ n.}$

Le système hybride consiste à garder pour le MSpec, la cellule C_k identifiée par les chaînes de Markov, et la cellule C_α correspond au résultat α obtenu par le module de prédiction.

Ainsi, pour améliorer les performances, la réservation à l'avance est faite uniquement dans les deux cellules C_k et C_α (si les chaînes de Markov et le module de prédiction identifient la même cellule, $C_k = C_\alpha$ est la réservation à l'avance est faite dans une seule cellule).

Le diagramme suivant représente la détermination du MSpec en utilisant les chaînes de Markov et le module de prédiction.

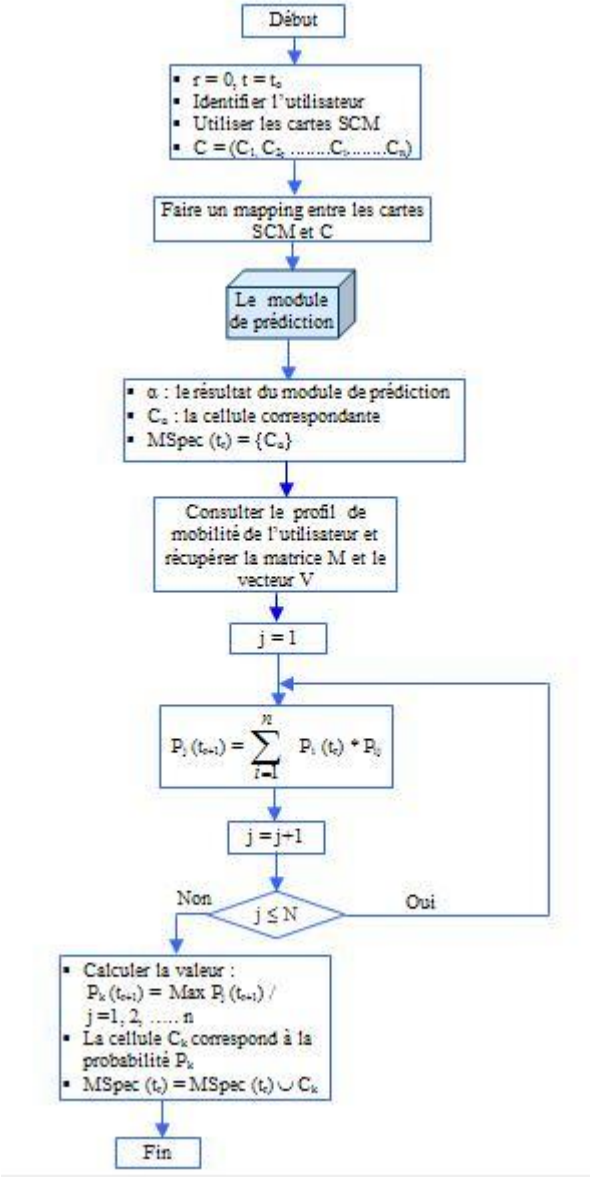


Figure 6. Prédire le MSpec par le système hybride

- *La décision de handover (extension du profil de mobilité pour la 4G)*

Cette décision est basée sur un mapping entre le type de chaque application et la localisation de l'utilisateur identifiée par le Cell ID.

Exemple: si l'utilisateur se trouve dans la cellule 1 et lance une application spécifique qui nécessite un niveau élevé de QoS (par exemple, une application de type vidéo), il est alors nécessaire de faire un handover vertical (de l'UMTS vers le WLAN), parce que la QoS nécessaire pour l'application ne peut pas être assurée dans un environnement UMTS. Ce paramètre représente une matrice DH.

DH [Cell ID, App ID] = { 0 : le handover vertical n'est pas nécessaire
1 : handover vertical vers UMTS (dans cet exemple)

1.2. MQoS NSLP entre deux terminaux mobiles

La section suivante décrit notre approche pour la réservation de ressources à l'avance dans un réseau sans fil, elle est basée sur l'application de signalisation QoS NSLP.

La réservation de ressources à l'avance pour l'utilisateur est faite uniquement dans les localisations où il peut se rendre, ces futures localisations sont déterminées après la phase d'observation, c'est-à-dire après la détermination du profil de mobilité de l'utilisateur.

Nous avons choisi une architecture HMIPv6 pour appliquer le modèle de réservation de ressources à l'avance, les raisons pour ce choix sont basées sur l'adaptation de l'architecture HMIPv6 à la réservation de ressources à l'avance et notamment l'existence de l'entité MAP *(Mobility Anchor Point)* qui joue un rôle très important lors de la réservation des ressources à l'avance pour le compte du terminal mobile et permet ainsi de réduire la signalisation globale dans le réseau qui est l'un des objectifs visé par cette procédure.

Dans ce qui suit, nous présentons MQoS NSLP (La procédure de réservation de ressources à l'avance dans un réseau mobile basée sur l'application de signalisation QoS NSLP) entre deux terminaux mobiles.

Le scénario suivant, présente MQoS NSLP dans le cas où le MH1 est l'entité qui génère le flux avec le mode *Sender Initiated Reservation.*

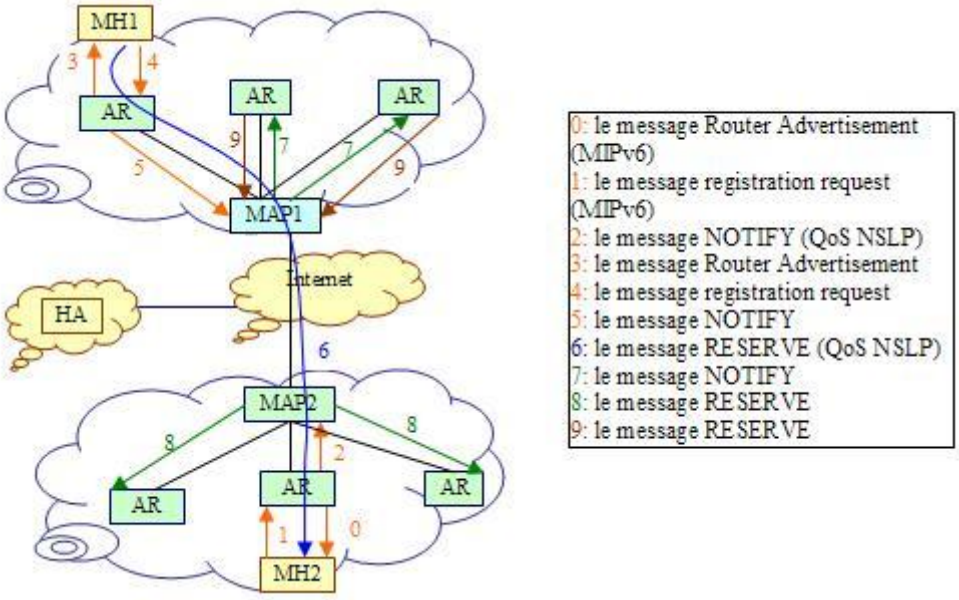


Figure 7. Réservation de ressources à l'avance avec les messages de QoS NSLP

La figure 7 représente l'ensemble des interactions entre les deux terminaux MH1 et MH2 afin de réserver les ressources à l'avance. Dans ce scénario, les deux entités qui communiquent sont mobiles, c'est le MH1 qui génère le flux. Le MH1 représente le NI, le MH2 représente le NR, les AR ainsi que les MAP représentent les entités NF.

On note MSpec1, MSpec2 les deux ensembles de cellules que le MH1 et le MH2 peuvent visiter durant l'association avec le système.

Dans un mode Sender Initiated Reservation, la procédure de réservation de ressources à l'avance avec QoS NSLP est la suivante (remarque : l'enregistrement peut commencer soit

avec le MH1 ou le MH2, le scénario suivant considère que le MH2 est le premier mobile qui fait l'enregistrement) :

0 : l'AR informe le MH2 avec le message *Router Advertisement* de la disponibilité de ressources à l'aide du bit Q.

Si Q = 0 alors l'AR ne possède pas de ressources et dans ce cas le MH2 ne peut se connecter qu'en Best Effort.

1 : pendant l'enregistrement le MH2 demande la QoS à son AR, en utilisant le bit Q dans le message *registration request. O*n propose d'ajouter l'objet MSpec2 au message *registration request.*

2 : après l'enregistrement avec le MH2, l'AR envoie la demande de QoS au MAP2. Pour cela, on utilise le message NOTIFY en incluant l'objet MSpec2.

Après la réception du message NOTIFY, le MAP2 analyse l'objet MSpec2 (il détermine la durée de la réservation ainsi que les futures localisations du MH2).

3 : l'AR informe le MH1 avec le message *Router Advertisement* de la disponibilité de ressources en utilisant le bit Q.

Si Q = 0 alors l'AR ne possède pas de ressources et dans ce cas le MH1 ne peut se connecter qu'en BE.

4 : pendant l'enregistrement le MH1 demande la QoS à son AR, en utilisant le bit Q dans le message *registration request. O*n propose d'ajouter l'objet MSpec1 au message *registration request.*

5 : après l'enregistrement avec le MH1, l'AR envoie la demande de QoS au MAP1, pour cela on utilise le message NOTIFY en incluant l'objet MSpec1. Après la réception du message NOTIFY, le MAP1 analyse l'objet MSpec1 (il détermine la durée de la réservation ainsi que les futures localisations du MH1).

6 : pour réserver les ressources entre le MH1 et le MH2, le MH1 (NI) envoie le message RESERVE qui doit contenir l'objet QSpec. Ce message est transporté par GIMPS jusqu'au MAP1, il sera par la suite envoyé au MAP2, puis envoyé à l'AR pour arriver finalement au MH2 (NR).

7 : le MAP1, après réception du message RESERVE, envoie le message NOTIFY à tous les AR qui se trouvent dans le MSpec1 pour les inciter à envoyer le message RESERVE.

8 : le message RESERVE est retransmis, après sa réception par le MAP2, à tous les AR qui se trouvent dans le MSpec2.

9 : les ARs se trouvant dans le MSpec1 répondent au MAP1 avec le message RESERVE.

1.2.1 La procédure de handover

Durant le handover, le MH1 et le MH2 passent par les étapes suivantes :

- MH2 s'enregistre auprès du nouvel AR (protocole MIPv6).
- Pour minimiser les pertes de paquets et avant l'installation du nouveau chemin, nous proposons de faire un transfert de contexte entre l'ancien et le nouvel AR en utilisant le protocole CTP (Context Transfer Protocol). Dans le cas d'un mode « Network controlled », initié par le nAR, les messages suivants sont utilisés :
  - Le nouvel AR envoie le message CT Request à l'ancien AR.
  - L'ancien AR répond en utilisant le message CTD (le message CTDR est optionnel).
- Etablissement du nouveau chemin et mise à jour de la réservation de ressources :
  - Le nouvel AR envoie le message RESERVE au MH2, (message 1 sur la figure 8).

- Le MH2 répond avec le message RESPONSE en incluant le nouveau MSpec2, (message 2 sur la figure 8).
- Après la réception du message RESPONSE, le nouvel AR envoie le message NOTIFY au MAP2 en incluant le nouveau MSpec2 (message 3 sur la figure 8).
- Le MAP2 analyse le nouveau MSpec2, et effectue, à travers le message RESERVE, les actions suivantes (message 4 sur la figure 8) :
    - Garder la réservation pour l'ancienne cellule si elle appartient au nouveau MSpec2, sinon supprimer la réservation.
    - Faire des réservations à l'avance dans les nouvelles cellules qui n'appartiennent pas à l'ancien MSpec2.
    - Supprimer la réservation pour les anciennes cellules qui n'appartiennent pas au nouveau MSpec2 sauf, bien sûr, pour la cellule courante.
- MH1 s'enregistre auprès du nouveau AR (protocole MIPv6).
- Pour minimiser les pertes de paquets et avant l'installation du nouveau chemin, nous proposons de faire un transfert de contexte entre l'ancien et le nouvel AR par l'intermédiaire du protocole CTP (Context Transfer Protocol). Dans le cas d'un mode « Network controlled », initié par le nAR, les messages suivants sont utilisés :
    - Le nouvel AR envoie le message CT Request à l'ancien AR.
    - L'ancien AR répond en utilisant le message CTD (le message CTDR est optionnel).
- Etablissement du nouveau chemin et mise à jour de la réservation de ressources :
    - Le MH1 envoie le message RESERVE au nouvel AR en incluant le nouveau MSpec1, il sera retransmis pour arriver au MAP1 (message 5 sur la figure 8).
    - Le MAP1 inclut l'ancien MSpec1 et le nouveau MSpec1 dans un message NOTIFY et l'envoie à tous les ARs identifiés par le nouveau et l'ancien MSpec1 (message 6 sur la figure 8).

Chaque AR analyse les deux objets MSpec1 et effectue, à travers le message RESERVE, les actions suivantes (message 7 sur la figure 8) :

- l'ancien AR garde la réservation pour l'ancienne cellule si elle appartient au nouveau MSpec1, sinon il supprime la réservation.
- chaque nouvel AR fait des réservations à l'avance dans les nouvelles cellules qui n'appartiennent pas à l'ancien MSpec1.
- chaque ancien AR Supprime la réservation pour les anciennes cellules qui n'appartiennent pas au nouveau MSpec1 sauf, bien sûr, pour la cellule courante.

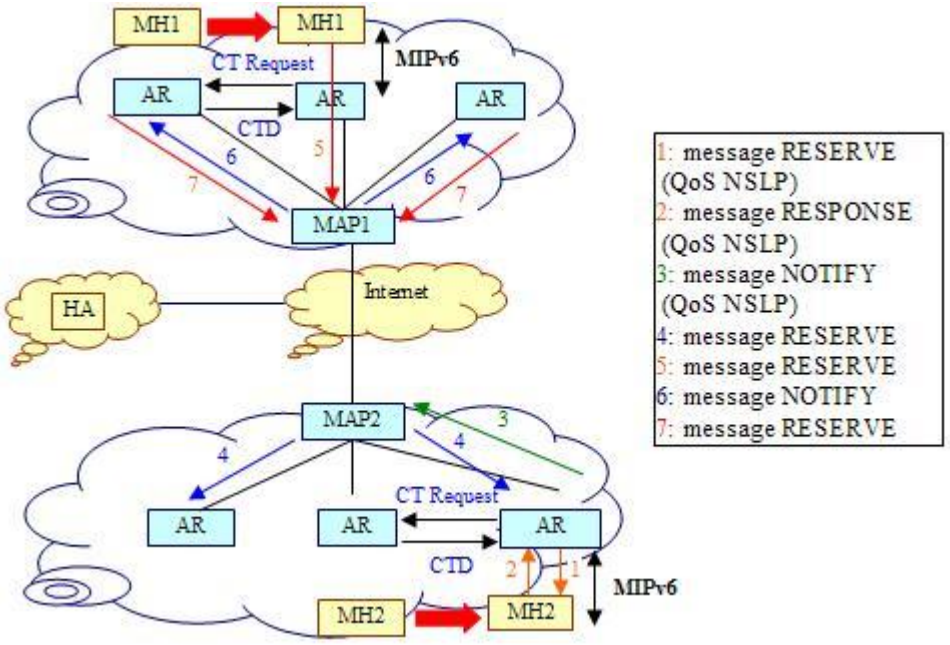


Figure 8. La procédure de handover

1.2.1.1. Les actions réalisées par l'entité MAP

Le diagramme suivant représente les actions réalisées par le MAP2 dans le cas d'une communication entre deux terminaux mobiles, le MAP2 reçoit comme information le MSpec2 envoyé par le MH2. Afin de modéliser les actions réalisées par le MAP2, la première cellule dans laquelle l'utilisateur se connecte pour la première fois est considérée comme C_1 et après chaque handover le mobile passe de C_i à C_{i+1}, ce qui veut dire que même si pour le $2^{ème}$ handover, le mobile retourne à C_1, pour le MAP2 cette cellule sera considérée comme C_3. Afin de mieux gérer les ressources réseaux, on utilise les deux types de réservation passive et active, une réservation pour la future cellule de l'utilisateur est toujours passive, elle devient active uniquement lorsque l'utilisateur se déplace vers cette cellule.

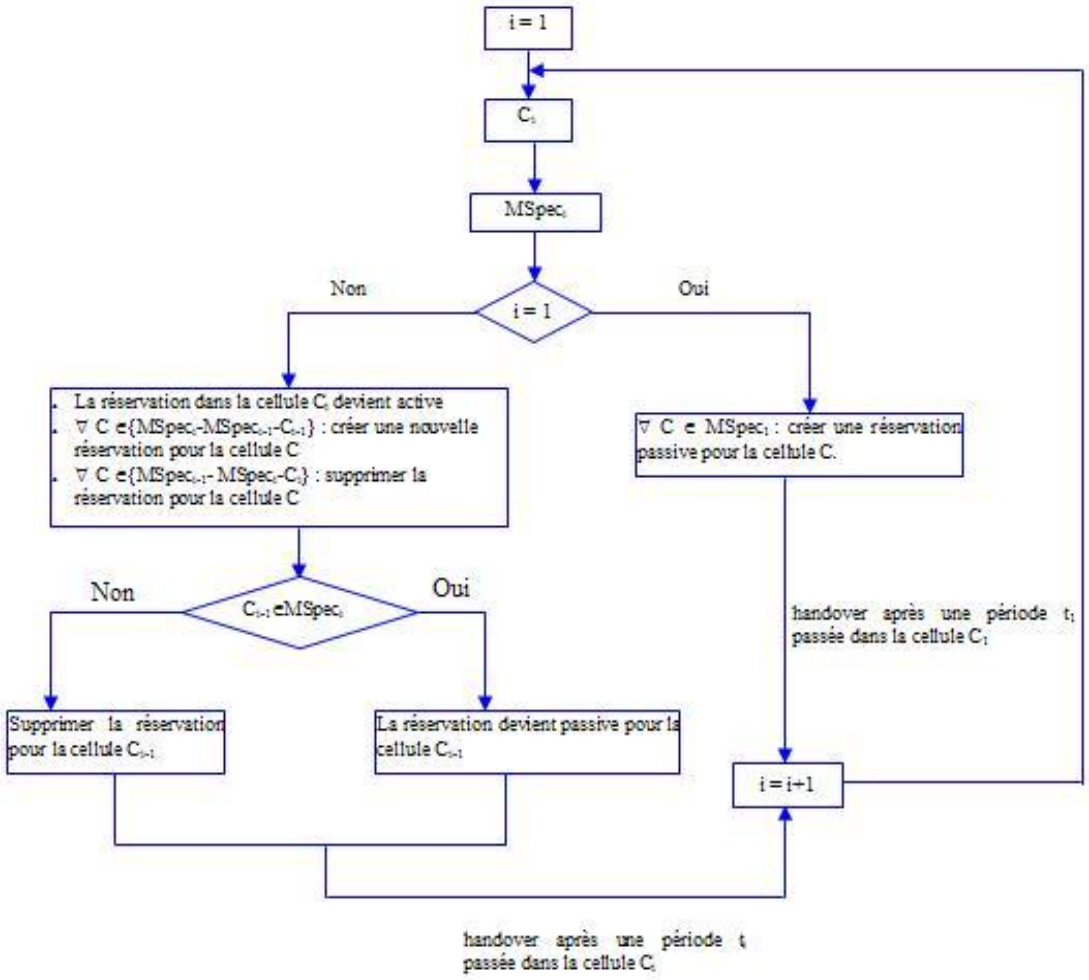


Figure 9. Les actions réalisées par le MAP2 dans le cas d'une communication entre deux terminaux mobiles

1.3. Scénario de référence pour la 4G

L'utilisateur de la 4ème génération de mobiles (4G) a plusieurs technologies d'accès sans fil à sa disposition. Cet utilisateur veut pouvoir être connecté au mieux, n'importe où, n'importe quand et avec n'importe quel réseau d'accès. Pour cela, les différentes technologies sans fil qui sont représentées sur la figure 10, doivent coexister de manière à ce que la meilleure technologie puisse être retenue en fonction du profil de l'utilisateur et de chaque type d'application qu'il demande.

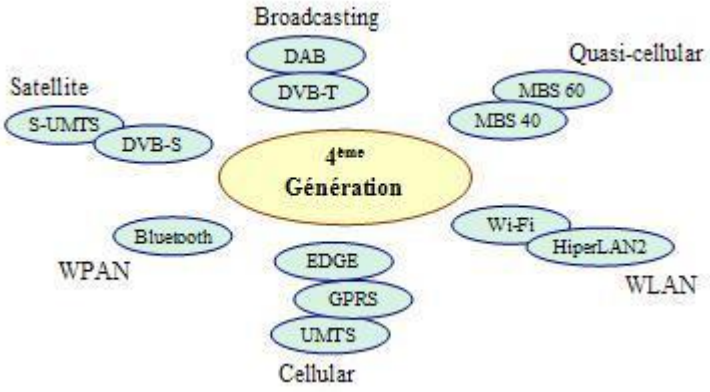


Figure 10. Plusieurs technologies d'accès pour l'utilisateur de la 4G

Dans ce contexte, l'équipement terminal devra rechercher en permanence le meilleur réseau d'accès en fonction des besoins de l'utilisateur. Dans notre scénario de référence, le terminal de l'utilisateur supporte deux technologies d'accès sans fil qui sont l'UMTS et le WLAN. Le profil de mobilité est, dans ce cas, utilisé pour adapter le handover vertical (inter-technologie) aux besoins de qualité de service de l'utilisateur.

Le tableau suivant montre, clairement, que les deux technologies UMTS et WLAN sont complémentaires :

WLAN	UMTS
• Déployé dans des environnements intérieurs (Indoor environment) • Petite zone de mobilité • Une mobilité lente • Débit très élevé • Faible coût de déploiement • Adapté aux hotspots	• Déployé dans des environnements extérieurs (Outdoor environment) • Large zone de mobilité • Grande mobilité • Débit modéré • Coût de déploiement élevé • Déconseillé pour les hotspots

Tableau 1. Comparaison entre l'UMTS et le WLAN

1.3.1 L'architecture d'intégration

Définir une architecture d'intégration entre l'UMTS et le WLAN présente plusieurs défis, comme le choix du meilleur point d'intégration entre les deux technologies, car le WLAN peut être connecté avec l'UMTS par le RNC, le SGSN, ou le GGSN [2].

Le RNC (Radio Network Controller) contrôle l'utilisation et l'intégrité des ressources radio dans l'UMTS. Choisir le RNC comme point d'intégration entre l'UMTS et le WLAN, nécessite un changement important au niveau des procédures radio implémentées au niveau du RNC, les interfaces radio étant totalement différentes entre l'UMTS et le WLAN. Le choix du GGSN (Gateway GPRS Support Node) comme point d'intégration simplifie le handover de l'UMTS vers le WLAN, car, dans ce cas, le GGSN maintient simplement la session pour la connexion PS (Packet Switched). Mais, lors d'un handover de l'UMTS vers le WLAN, le SGSN (Serving GPRS Support Node) à besoin de créer l'état de la mobilité (mobility state), et de rétablir la session PDP (Packet Data Protocol) ainsi que le contexte RAB (Radio Access Bearer), des traitements que le GGSN ne peut pas faire, ce qui ralentira le handover.

Pour ces raisons, nous avons choisi le SGSN comme point d'intégration entre l'UMTS et le WLAN. La figure 11 montre la connexion du WLAN au SGSN à travers l'entité MAP.

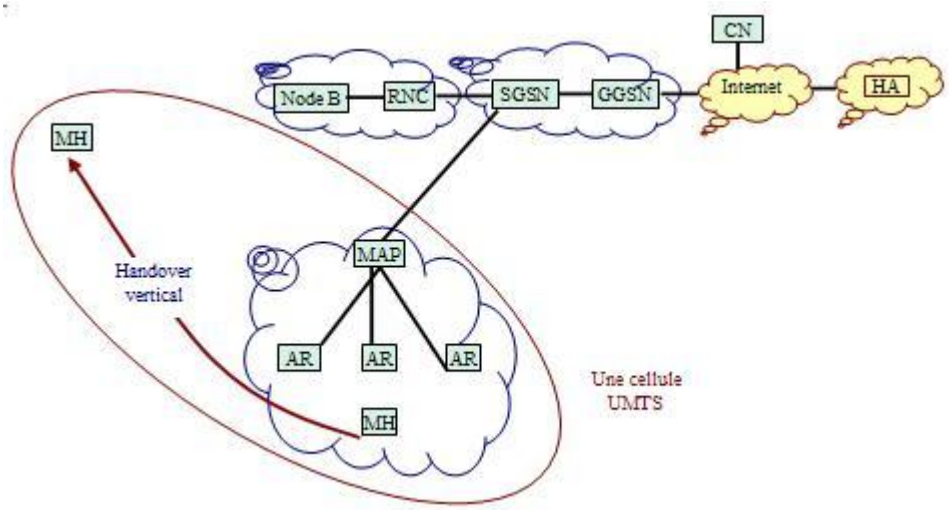


Figure 11. L'architecture d'intégration

On suppose que l'utilisateur lance une seule application à la fois.
Cell ID : identifie la cellule courante.
App ID : identifie l'application en cours d'exécution.
App ID1: identifie la première application lancée par l'utilisateur.
type_application : variable qui retourne le type de chaque application.
technologie-accès = WLAN

La procédure retenue est la suivante :

Consulter le profil de mobilité pour l'utilisateur, afin de récupérer la matrice DH.
Si DH [Cell ID, App ID] = 0 **alors** garder la connexion WLAN
Sinon faire un handover vertical vers l'UMTS
Fin Si
Après le handover vertical, technologie-accès = UMTS
Tant que type_application = App ID1 **Faire** garder la connexion UMTS
(Après la fermeture de l'application en cours et le lancement d'une nouvelle application).
Consulter la matrice DH.
Si DH [Cell ID, App ID] = 1 **alors** garder la connexion UMTS
Sinon faire un handover vertical vers le WLAN
Fin Si

1.4. Modèle mathématique, simulation avec MATLAB

L'objectif de cette simulation est de trouver la bonne valeur de θ afin de simuler le système ainsi que le calcul du temps nécessaire à la détermination du MSpec.
L'élément clé pour la réservation de ressources à l'avance est l'ensemble MSpec.
Le MSpec est défini comme suit : MSpec $(t_r) = \{C_j / P_j (t_{r+1}) \geq \theta \}$, $(0 \leq \theta \leq 1)$.

Le choix de la valeur de θ est très important pour construire le MSpec. Plus θ est grand (proche du 1), plus le MSpec est réduit, si θ est assez petit (proche du 0), l'ensemble MSpec contient tout le voisinage.

On définit la fonction f suivante : $f(a_i, C_i) = \begin{cases} \{\} & \text{si} \quad a_i = 0 \\ \{C_i\} & \text{si} \quad a_i = 1 \end{cases}$

Le MSpec est défini par : MSpec = $\bigcup_{i=1}^{n} f(a_i, C_i)$.

Les paramètres suivants sont liés au MSpec et sont considérés pour la simulation :

Nb_cel_MSpec: le nombre de cellules constituant le MSpec.

T_Détermination: le temps de la détermination du MSpec par le terminal mobile.

Taux_échec: le taux d'échec pour la détermination du MSpec.

Nb_cel_MSpec: le nombre de cellules constituant le MSpec.

- Synthèse mathématique :

Nous avons : C = (C_1, C_2, ...C_i.....C_n) : l'ensemble des cellules dans le voisinage du terminal mobile.

Le MSpec est défini comme suit : MSpec (t_r) = {C_j / P_j (t_{r+1}) ≥ θ }, (0 ≤ θ ≤ 1).

Si θ = 0 alors le MSpec = C (l'ensemble MSpec contient tout le voisinage), et le nombre de cellules constituant le MSpec est n.

Si θ = 1 alors le MSpec = {} (l'ensemble MSpec est vide). Dans ce cas le nombre de cellules constituant le MSpec est 0.

La fonction f est définit comme suit : $f(a_i, C_i) = \begin{cases} \{\} & \text{si } a_i = 0 \\ \{C_i\} & \text{si } a_i = 1 \end{cases}$

Et par conséquent, on peut en déduire la formule suivante :

MSpec = $\bigcup_{i=1}^{n} f((1-\theta), C_i)$.

On a : 0 ≤ θ ≤ 1 et donc 0 ≤ 1-θ ≤ 1.

Pour la suite on considère que (1-θ) représente le facteur qui détermine le sous ensemble de C. par exemple si (1-θ = 0.25), on déduit que le MSpec contient 25% des éléments de C.

Formellement on a : Card (MSpec) = E ((1-θ) * Card (C)) = E (n * (1-θ)).

Card : définit la cardinalité d'un ensemble qui est le nombre d'éléments de l'ensemble.

E : est la fonction partie entière, elle est définie pour tout réel x de la façon suivante : E(x) est l'entier relatif immédiatement inférieur ou égal à x.

Cependant si θ = 0.5, on n'est pas sûr que le MSpec contienne la moitié des cellules de C. Donc au lieu de prendre le facteur (1-θ), on prendra le facteur $(1-\theta)^k$, avec un k réel positif.

On a donc : Card (MSpec) = E (n * $(1-\theta)^k$).

Si 0 < k < 1 **alors** (1-θ) < $(1-\theta)^k$: dans ce cas, avec $(1-\theta)^k$, le MSpec contient plus de cellules.

Exemple : **si** n = 20, **k = 0.5**, 1-θ = 0.5 **alors** Card (MSpec) = 14.

Si k > 1 **alors** (1-θ) > $(1-\theta)^k$: dans ce cas, avec $(1-\theta)^k$, le MSpec contient moins de cellules.

Exemple : **si** n = 20, **k = 2**, 1-θ = 0.5 **alors** Card (MSpec) = 5.

Le nombre de cellules constituant le MSpec est alors le suivant :

Nb_cel_MSpec = E (n * $(1-\theta)^k$).

Et on a $(1-\theta)^k$: c'est le facteur qui détermine le sous ensemble de C.

- Simulation avec MATLAB :

L'emplacement des cellules les unes par rapport aux autres influence les probabilités de transition. Nous considérons l'emplacement suivant des cellules, en supposant que le voisinage de l'utilisateur contient 10 cellules.

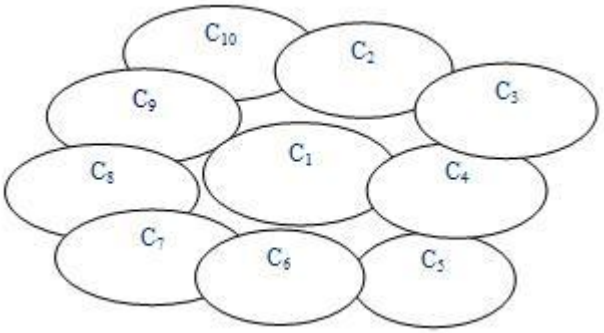


Figure 12. L'emplacement des cellules dans le voisinage du terminal mobile

On remarque que si le terminal mobile s'associe avec le point d'accès qui se trouve dans la cellule C_2 par exemple, il ne peut pas se déplacer vers la cellule C_6 après le handover sans passer par la cellule C_1 ou bien passer par C_3, C_4, C_5 ou n'importe quel autre chemin mais jamais directement de C_2 à C_6.

On suppose que m = 40 (c'est le nombre d'associations avec le système durant la phase d'observation).

Durant les 40 associations avec le système, on va suivre les différents trajets du terminal mobile durant la durée de vie d'un flux. Un trajet est une suite de cellules qui marque le chemin suivi par le terminal mobile.

Exemple : C_4, C_1, C_7, C_6, C_1 est un trajet pour le terminal mobile.

Donc on va définir un ensemble de trajets afin de déterminer la matrice M et le vecteur V.

On calcule k (i) de i = 1 à 10.

k (i) : c'est le nombre de fois que l'utilisateur se connecte dans la cellule i à l'instant t_o. On a donc le tableau suivant :

k (1)	k (2)	k (3)	k (4)	k (5)	k (6)	k (7)	k (8)	k (9)	k (10)
0	5	7	8	0	12	0	8	0	0

Par la suite, on a le vecteur V qui contient les P_i (t_o), il est calculé à l'aide de la formule suivante :

P_i (t_o) = k (i) / m.

P_1 (t_o)	P_2 (t_o)	P_3 (t_o)	P_4 (t_o)	P_5 (t_o)	P_6 (t_o)	P_7 (t_o)	P_8 (t_o)	P_9 (t_o)	P_{10} (t_o)
0	0.125	0.175	0.2	0	0.3	0.075	0.2	0	0

Avant de calculer la matrice M, on calcule la matrice qui contient les valeurs t [i, j].

t [i, j] : le nombre de transitions de la cellule i vers la cellule j pendant les **40** associations avec le système.

La matrice qui contient les valeurs t [i, j] est la suivante :

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
C_1	0	6	6	4	3	2	3	2	3	2
C_2	5	0	7	0	0	0	0	0	0	4
C_3	5	9	0	8	0	0	0	0	0	0
C_4	5	0	8	0	7	0	0	0	0	0
C_5	4	0	0	5	0	5	0	0	0	0
C_6	6	0	0	0	8	0	6	0	0	0
C_7	3	0	0	0	0	4	0	3	0	0
C_8	3	0	0	0	0	0	3	0	5	0
C_9	2	0	0	0	0	0	0	3	0	3
C_{10}	3	4	0	0	0	0	0	0	2	0

g (i) : le nombre de transitions qui ont comme point de départ la cellule i pendant les **40** associations. On le calcule de la manière suivante : g (i)= $\sum_{j=1}^{10}$ t [i, j]. On obtient le tableau suivant :

g (1)	g (2)	g (3)	g (4)	g (5)	g (6)	g (7)	g (8)	g (9)	g (10)
31	16	22	20	14	20	10	11	8	9

Enfin, on calcule la matrice M qui contient les P [i, j] avec la formule P [i, j] = t [i, j] / g (i).
La matrice M est la suivante :

P	1	2	3	4	5	6	7	8	9	10
1	0	0.19	0.19	0.14	0.1	0.06	0.1	0.06	0.1	0.06
2	0.31	0	0.44	0	0	0	0	0	0	0.25
3	0.22	0.4	0	0.38	0	0	0	0	0	0
4	0.25	0	0.4	0	0.35	0	0	0	0	0
5	0.3	0	0	0.35	0	0.35	0	0	0	0
6	0.3	0	0	0	0.4	0	0.3	0	0	0
7	0.3	0	0	0	0	0.4	0	0.3	0	0
8	0.27	0	0	0	0	0	0.27	0	0.46	0
9	0.25	0	0	0	0	0	0	0.375	0	0.375
10	0.33	0.44	0	0	0	0	0	0	0.23	0

Le vecteur V est le suivant :

$P_1(t_o)$	$P_2(t_o)$	$P_3(t_o)$	$P_4(t_o)$	$P_5(t_o)$	$P_6(t_o)$	$P_7(t_o)$	$P_8(t_o)$	$P_9(t_o)$	$P_{10}(t_o)$
0	0.125	0.175	0.2	0	0.3	0	0.2	0	0

Après les **40** associations avec le système, la matrice **M** et le vecteur **V** sont fixés. Le système calcule le vecteur $V_1 = V * M$ afin de construire le MSpec1 pour le 1er handover. Pour le 2ème handover le système calcule le vecteur $V_2 = V_1 * M$ afin de construire le MSpec2 et ainsi de suite, pour le Ième handover le système calcule le vecteur $V_i = V_{i-1} * M$.

A l'aide de MATLAB et après 6 handovers, on a les résultats suivants :

V_1= [0.2712 0.0700 0.1350 0.0665 0.1900 0 0.1440 0 0.0920 0.0313]

V_2= [0.2015 0.1193 0.1089 0.1558 0.0504 0.1404 0.0271 0.0940 0.0343 0.0683]

V_3= [0.2217 0.1119 0.1531 0.0873 0.1308 0.0406 0.0876 0.0331 0.0791 0.0548]

V_4= [0.2147 0.1275 0.1263 0.1350 0.0689 0.0941 0.0433 0.0693 0.0500 0.0709]

V_5= [0.2176 0.1225 0.1509 0.1022 0.1064 0.0543 0.0684 0.0446 0.0696 0.0635]

V_6= [0.2159 0.1296 0.1361 0.1250 0.0792 0.0777 0.0501 0.0597 0.0569 0.0698]

Le graphe suivant montre l'impact de la valeur de θ sur le nombre de cellules constituant le MSpec.

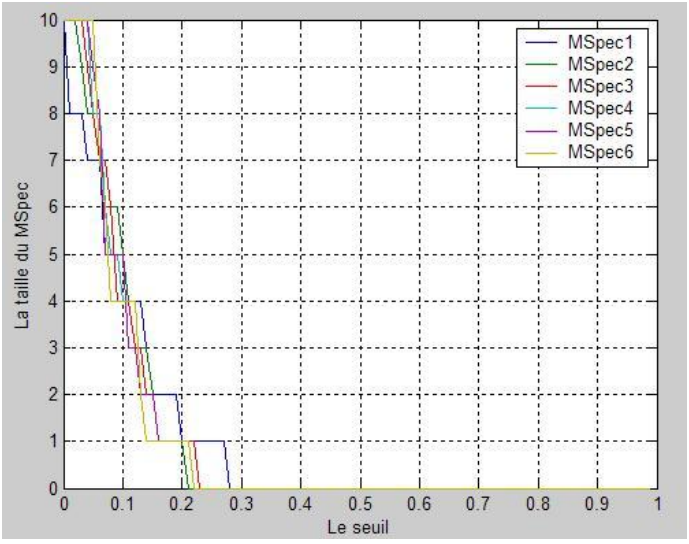


Figure 13. L'impact de la valeur de θ sur le nombre de cellules constituant le MSpec.

On remarque que à partir de la valeur 0.28 de θ, le MSpec est vide, donc des valeurs de θ supérieures à 0.28 ne sont pas intéressantes pour la simulation. Pour ce qui suit et pour le reste de la simulation, on va prendre θ = 0.1.

Avec θ = 0.1, nous avons les résultats suivants :

MSpec 1 = {C_1, C_3, C_5, C_7}, Nb_cel_MSpec=4.

MSpec 2 = {C_1, C_2, C_3, C_4, C_6}, Nb_cel_MSpec=5.

MSpec 3 = {C_1, C_2, C_3, C_5}, Nb_cel_MSpec=4.

MSpec 4 = {C_1, C_2, C_3, C_4}, Nb_cel_MSpec=4.

MSpec 5 = {C_1, C_2, C_3, C_4, C_5}, Nb_cel_MSpec=5.

MSpec 6 = {C_1, C_2, C_3, C_4}, Nb_cel_MSpec=4.

T_Détermination : le temps de la détermination du MSpec.

Le MSpec est déterminé par le terminal mobile de l'utilisateur.

Le graphe suivant montre l'impact de la valeur de θ sur le temps de la détermination du MSpec. (Le temps est calculé en milliseconde).

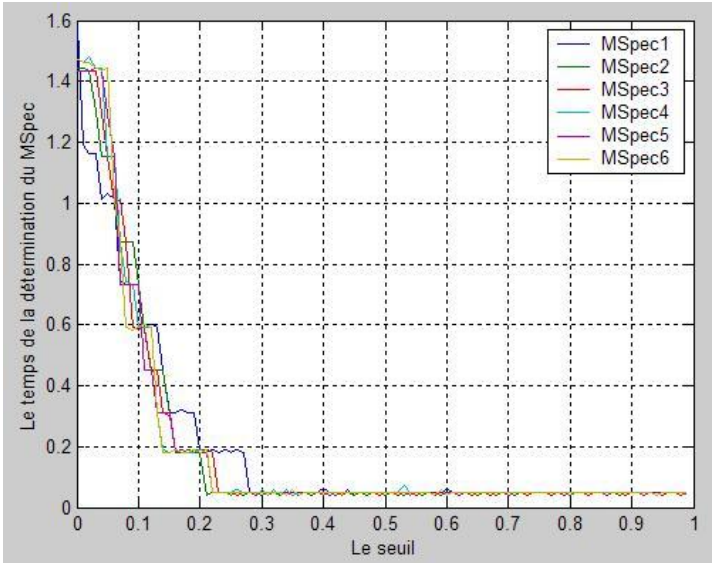


Figure 14. L'impact de la valeur de θ sur le temps de la détermination du MSpec

Ces performances sont calculées sur une machine qui a une puissance de 1,4 GHZ/s. Le temps de calcul peut varier selon la puissance de la machine, dans le cas général, on peut calculer le nombre de coups d'horloge pour une opération donnée :

Exemple :

Un calcule de 1,4 ms (milliseconde) sur une machine qui a une puissance de 1,4 GHZ/s, nécessite $1,4 * 10^{-3} * 1,4 * 10^{9} = 1,96 * 10^{6}$ coups d'horloge.

Taux_échec : le taux d'échec pour la détermination du MSpec.

Le taux d'échec pour la détermination du MSpec est calculé après chaque trajet.

Exemple :

Avec θ = 0.1, nous avons les résultats suivants (l'utilisateur se trouve dans la cellule C_{10}) :

MSpec 1 = {C_1, C_3, C_5, C_7}, MSpec 2 = {C_1, C_2, C_3, C_4, C_6}, MSpec 3 = {C_1, C_2, C_3, C_5}, MSpec 4 = {C_1, C_2, C_3, C_4}, MSpec 5 = {C_1, C_2, C_3, C_4, C_5}, MSpec 6 = {C_1, C_2, C_3, C_4}.

Les 6 ensembles MSpec, déterminent les futures localisations prédits par le système durant les 6 handovers pendant la durée de vie d'un flux.

Pour déterminer le taux d'échec du MSpec et pour un flux qui dure 6 handovers, on considère le trajet le plus suivi par l'utilisateur durant la phase d'observation et qui dure 6 handovers, c'est le trajet suivant : C_{10}, C_1, C_6, C_7, C_1, C_5, C_6.

On remarque que sur les 6 handovers, l'échec pour le calcul du MSpec, concernait le MSpec3 et le MSpec6, pour le $3^{ème}$ handover l'utilisateur s'est dirigé vers la cellule C_7 qui n'appartient pas au MSpec3 et pour le $6^{ème}$ handover l'utilisateur s'est dirigé vers la cellule C_6 qui n'appartient pas au MSpec6. Pour ce trajet le Taux_échec = 2/6 = 33.33%.

2. L'approche Agent

Dans le cas d'un nouvel utilisateur, le système ne peut pas lui réserver les ressources à l'avance car son profil de mobilité est jusqu'à présent inconnu et l'ensemble de ses futures localisations ne peut pas être calculé, on fait alors appel à la technologie Agent afin d'améliorer la qualité de service pour l'utilisateur mobile. Les caractéristiques des agents en terme de coopération, coordination et communication sont utilisées afin de satisfaire les besoins de l'utilisateur mobile en terme de QoS. Le terminal mobile qui est considéré dans ce cas est un terminal qui supporte deux technologies d'accès, à savoir Wi-Fi et UMTS, le rôle des agents dans ce cas est d'adapter le handover horizontal et le handover vertical aux besoins de QoS de l'utilisateur.

Dans le cadre de la 4ème génération de mobile (4G), l'utilisateur a plusieurs technologies d'accès sans fil à sa disposition. Cet utilisateur veut pouvoir être connecté au mieux, n'importe où, n'importe quand et avec n'importe quel réseau d'accès. Pour cela, les différentes technologies sans fil qui sont représentées, dans la figure 15, doivent coexister de manière à ce que la meilleure technologie puisse être retenue en fonction du profil de l'utilisateur et de chaque type d'application et de service qu'il demande.

Dans ce contexte, l'équipement terminal devra rechercher en permanence le meilleur réseau d'accès en fonction des besoins de l'utilisateur. La technologie agent peut jouer un rôle très important dans ce choix.

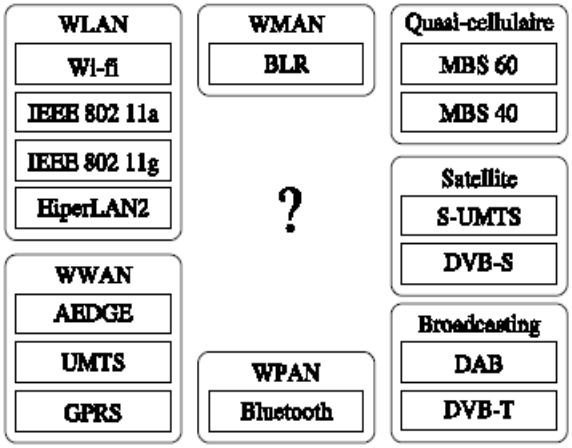


Figure 15. Les différentes technologies d'accès sans fil pour l'utilisateur 4G

2.1. Le modèle Agent

Il y a peu de travaux basés sur les systèmes multi agent dans le domaine des réseaux mobiles. La figure 16 donne un exemple de configuration réseau offrant deux technologies d'accès à l'utilisateur, à savoir Wi-Fi et UMTS. La technologie Agent est utilisée pour adapter le handover horizontal (changement au sein d'une même technologie d'accès) et vertical (changement de technologie d'accès) aux besoins de QoS de l'utilisateur. L'exemple suivant permet d'illustrer ce principe.

Exemple :

Nous prenons comme exemple un réseau Wi-Fi déployé sur un campus (une université) et où la technologie Agent est utilisée pour gérer dynamiquement la mobilité de l'utilisateur.

Dans un environnement Wi-Fi, la stratégie de changement de point d'accès nécessite 4 étapes :

- la découverte d'un point d'accès cible,
- la synchronisation avec le point d'accès,
- l'envoi d'une authentification,
- l'établissement de l'association.

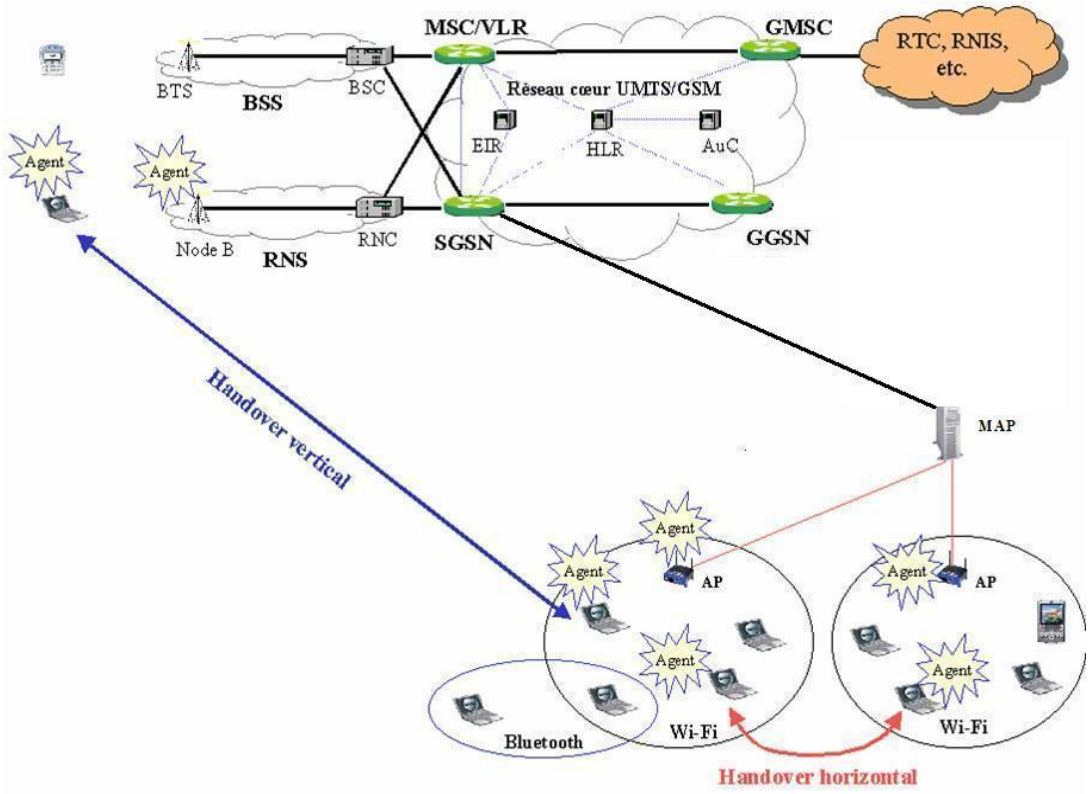


Figure 16. Handover horizontal et handover vertical

Cette stratégie, implémentée dans les équipements réseaux (les AP et les MH), est statique, c'est-à-dire que ni le fournisseur de service, ni le client ne peut changer la sélection du point d'accès. Pourtant cette sélection, dans certains cas, peut s'avérer mauvaise. Dans la figure 17, le Mobile Host 5 (MH5) sur lequel l'utilisateur lance une application gourmande en terme de QoS (une application vidéo par exemple), reçoit le meilleur signal de AP2, par contre la cellule 2 est déjà très chargée et par conséquent la QoS nécessaire pour MH5 ne peut pas être assurée. Une stratégie dynamique consiste à guider le MH5 vers la cellule 1 qui est vide et qui peut lui fournir la QoS nécessaire.

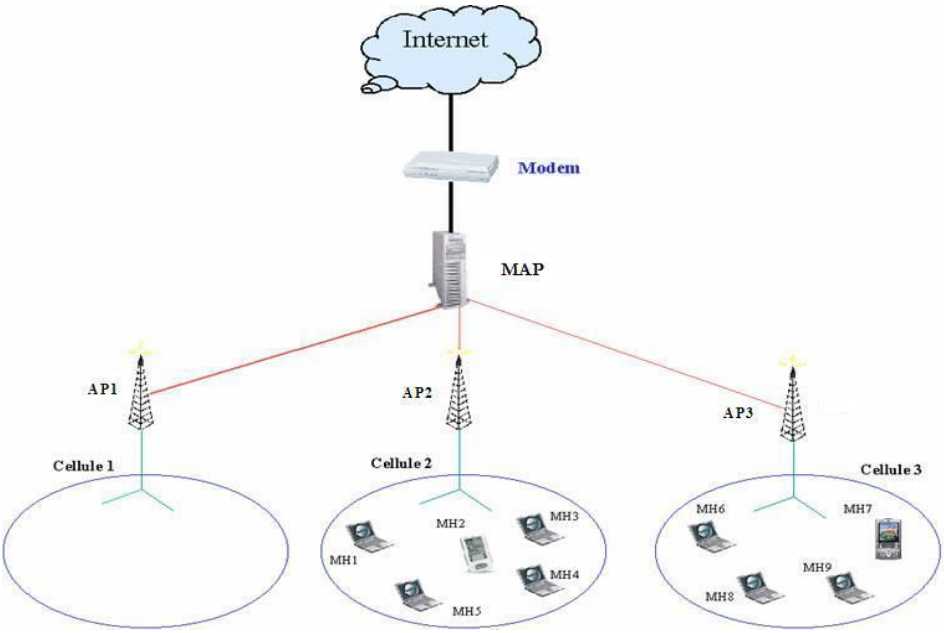


Figure 17. Exemple de réseau Wi-Fi

Si toutes les cellules sont remplies, l'utilisateur doit pouvoir utiliser une autre technologie d'accès à sa disposition répondant à ses besoins. Dans cet exemple, c'est la technologie UMTS qui sera utilisée, le grand nombre d'utilisateurs sur place empêchant le réseau Wi-Fi de répondre aux exigences de QoS de l'application demandée. Dès que l'utilisateur lance une autre application moins critique en terme de QoS ou que les performances du Wi-Fi deviennent acceptables pour l'application, l'utilisateur doit également pouvoir revenir sur la technologie Wi-Fi (à cause du coût élevé de l'UMTS par exemple). Les différents handovers verticaux doivent s'effectuer de manière totalement transparente pour l'utilisateur et en fonction des contraintes applicatives et du profil utilisateur.

Pour fournir de la qualité de service sur un lien Wi-Fi, il faut respecter 3 principes [3] :

- Le nombre d'hôtes autorisés à utiliser le canal doit être limité ;

- La zone géographique à l'intérieur de laquelle les utilisateurs communiquent doit être limitée de telle sorte qu'ils puissent tous utiliser le débit le plus élevé ;

- Les sources doivent être contraintes en configurant des conditionneurs de trafic dans les équipements.

Afin de fournir la QoS nécessaire à une application multimédia, nous allons respecter ces trois principes et faire trois suppositions :

- A partir d'un certain nombre d'utilisateurs (N), regroupés dans une même cellule, la QoS nécessaire pour une application multimédia ne sera plus assurée et la cellule sera considérée comme remplie.
- Chaque point d'accès contient un « identificateur de localisation » unique. A partir de cet identificateur de localisation, un utilisateur peut se connecter à la cellule permettant d'assurer la QoS nécessaire à l'application.
- En tenant compte du travail qui a été réalisé dans [4], une estimation de la position du MH est faite et une application donnant la répartition des cellules dans chaque salle de l'université (salle de conférence, bibliothèque,…) est téléchargeable à partir du serveur.

La figure 18 représente la répartition des cellules et les identificateurs associés, dans une salle de conférence possédant 3 points d'accès Wi-Fi.

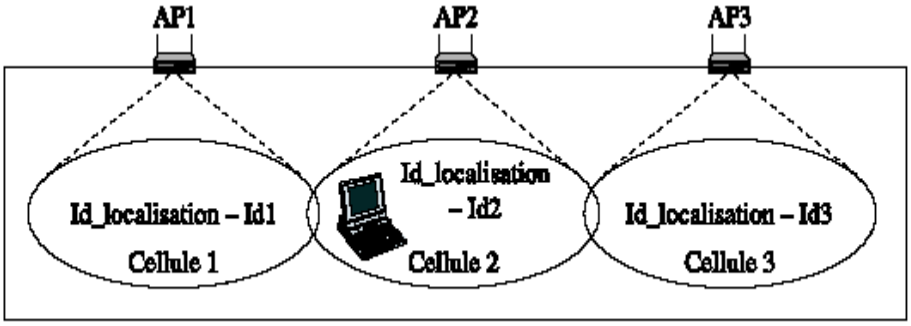


Figure 18. Distribution des cellules dans la salle

Le système multi-agents, contient trois agents :

- **Agent Application** : cet agent se situe sur le terminal mobile de l'utilisateur, son rôle est d'associer un profil applicatif à l'utilisateur, dans une première étape, l'agent **Application** détermine le type de l'application lancé par l'utilisateur, il l'associe à une classe de service selon le tableau suivant :

La classe de service	Type d'application
La classe 1	- Multimédia interactive. - Vidéo distribution. - Vidéo conférence. - Audio compressé. - Applications critiques en temps de réponse. - Transfert de données critiques (communications entre banques).
La classe 2	- Texte/données/transfert d'images. - Messagerie.

Tableau 2. Association classe de service et type d'application

Nous avons défini deux classes de service, la classe 1 représente les applications critiques en terme de QoS comme les applications multimédia interactives ou la vidéo conférence, la classe 2 représente les applications qui ne sont pas exigentes en terme de QoS comme la messagerie électronique.

Lors de l'étape suivante et pour chaque classe de service, l'agent application associe un profil applicatif à l'utilisateur.

Exemple : Pour l'application X qui appartient à la classe 1, l'utilisateur a pour profil : profil_applicatif_1 et pour une application qui appartient à la classe 2, l'utilisateur a pour profil : profil_applicatif_2. Pour ce dernier profil, l'utilisateur n'a aucun droit sur les applications et il ne peut pas lancer l'approche Agent, cette approche est lancée seulement lorsque l'utilisateur à comme profil applicatif le profil_applicatif_1.

- **Agent Terminal** : cet agent se situe sur le terminal mobile de l'utilisateur ; il fait la liaison entre l'utilisateur et le système ; il peut posséder une interface graphique ou en mode texte. Il communique avec l'agent **Etat** qui se trouve sur le point d'accès afin de connaître l'état de la

cellule ainsi que celle des cellules voisines. Il demande le déploiement d'une autre technologie d'accès si nécessaire. L'agent **Terminal** est activé par l'agent **Application**, après l'identification du profil applicatif de l'utilisateur comme par exemple profil_applicatif_1.

- **Agent Etat** : cet agent se situe sur le point d'accès ; il détermine l'état interne de la cellule ainsi que celle des cellules voisines. A partir de **N** utilisateurs regroupés dans la cellule, l'état de cette dernière sera considéré comme rempli. Pour connaître l'état des cellules voisines, l'agent **Etat** contacte les mêmes agents sur les cellules voisines, et peut ainsi récupérer leurs états.

2.2. Les interactions entre agents

La figure 19 représente l'ensemble des interactions dans le système. Dans cet exemple, l'utilisateur se trouve dans la cellule numéro 2 de la salle de conférence. Il consulte, par exemple, ses emails ou bien fait un transfert de fichiers. Dans ce cas, l'agent **Application** donne le profil_applicatif_2 à l'utilisateur, Au moment du lancement d'une application de la classe 1 (application multimédia, par exemple), l'agent **Application** donne le profil_applicatif_1 à l'utilisateur et active l'agent **Terminal (**message **m1),** ce dernier envoie un message (**m2**) à l'agent **Etat** afin de connaître l'état de la cellule courante. L'agent **Etat** compare le nombre d'utilisateurs dans la cellule avec le nombre N, et si ce dernier est inférieur ou égal au nombre d'utilisateurs dans la cellule, il envoie un message (**m3**) à l'agent **Terminal** pour lui indiquer que la cellule courante est remplie. Au même moment l'agent **Etat** contacte les mêmes agents sur les points d'accès voisins (messages **m4** et **m5**) pour connaître l'état des cellules voisines. Chaque agent répond par un message qui contient l'état de la cellule ou bien le nombre d'utilisateurs dans la cellule avec l'identificateur de localisation de la cellule (messages **m6** et **m7**). L'agent **Etat** dans la cellule courante fait une comparaison entre le nombre d'utilisateurs dans les cellules voisines ou bien entre leurs états et s'il existe au moins une cellule qui n'est pas remplie. Il envoie l'identificateur de localisation de la cellule choisie à l'agent **Terminal** (**m8**). Par la suite, le message **m9** qui est un message d'acquittement (ACK) est envoyé par l'agent **Etat** dans la cellule courante avec la cellule choisie.

A partir de ce moment, l'agent **Terminal** envoie une requête au serveur pour télécharger l'application qui lui permettra de savoir où se trouve la cellule concernée dans la salle et d'en informer l'utilisateur.

Si toutes les cellules sont remplies, l'agent **Etat** contacte l'agent **Terminal** pour lui communiquer la situation (message **m10**) et l'agent **Terminal** demandera le déploiement de l'UMTS.

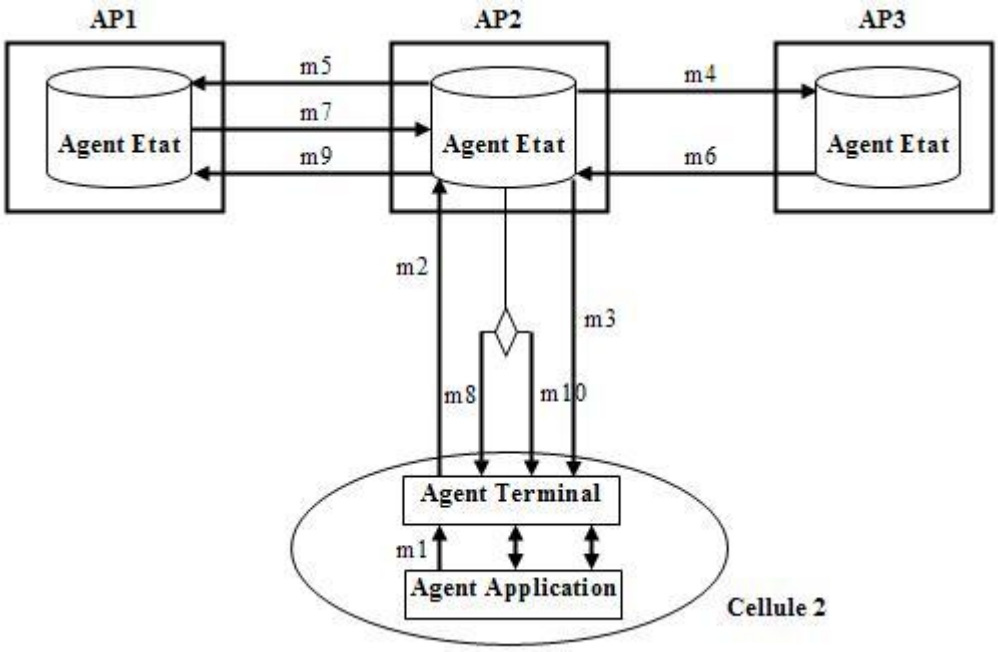


Figure 19. Les interactions entre les agents

2.3. Modélisation AUML des interactions dans le système

AUML (Agent Unified Modelling Language) est l'adaptation des notations UML pour décrire la modélisation orientée agent. Les modifications proposées sont :

- le soutien pour la représentation des *threads* simultanés d'interaction (comme la transmission de messages à plusieurs agents) permettant ainsi à UML de modéliser les protocoles d'interactions entre agents;

- la notion de rôle qui étend celle fournie par UML et permet de modéliser un agent qui joue plusieurs rôles.

La figure suivante représente l'ensemble des interactions dans le système modélisé en AUML.

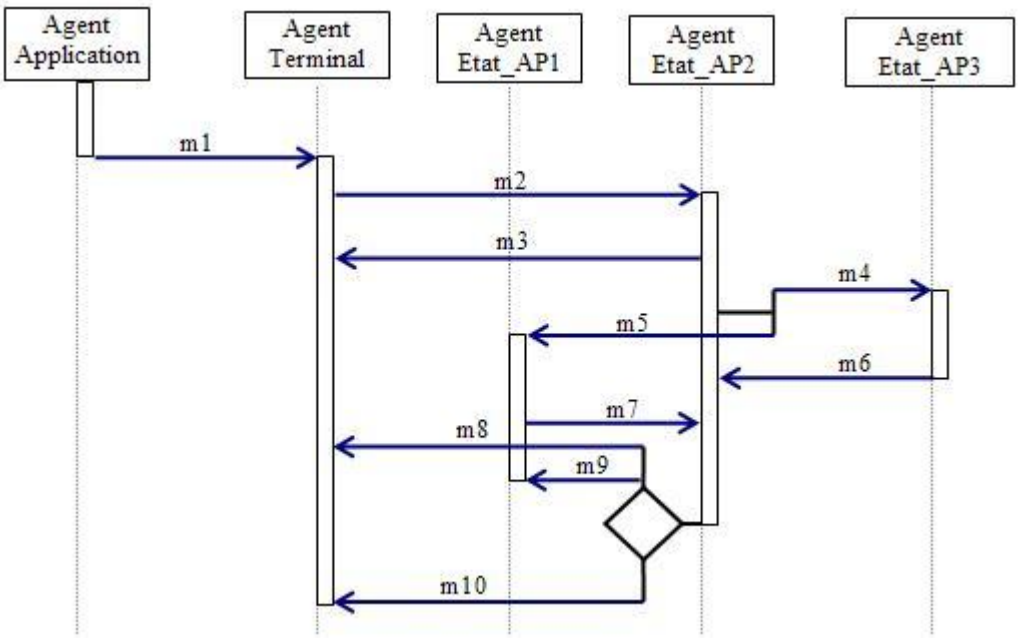


Figure 20. Modélisation AUML des interactions entre agents

2.4. Modélisation des interactions dans le système par un réseau de Pétri ordinaire

Un Réseau de Pétri est un outil de vérification et de validation pour les systèmes distribués, il représente un graphe orienté comportant :

- un ensemble fini de places, P= {P_1, P_2, P_3, ..., P_m}, symbolisées par des cercles et représentant des **conditions** :

- un ensemble fini de transitions, T= {T_1, T_2, T_3, ..., T_n}, symbolisées par des tirets et représentant l'ensemble des **événements** (les actions se déroulant dans le système) dont l'occurrence provoque la modification de l'état du système,

- un ensemble fini d'arcs orientés qui assurent la liaison d'une place vers une transition ou d'une transition vers une place,

Deux situations sont à vérifier dans un réseau de Pétri :

- La situation de conflit : dans un réseau de Pétri, des transitions sont en conflit structurel, si et seulement si elles ont au moins une place commune en entrée. Il y a conflit effectif s'il y a existence de conflit structurel et si pour un marquage Mi, le nombre de marques dans la place d'entrée est inférieur au nombre de transitions validées pour ce marquage.

- La situation de blocage : un réseau de Pétri, dans un marquage donné M, est en situation de blocage si et seulement si aucune transition n'est franchissable.

La figure suivante représente l'ensemble des interactions dans le système modélisées par un réseau de Pétri ordinaire.

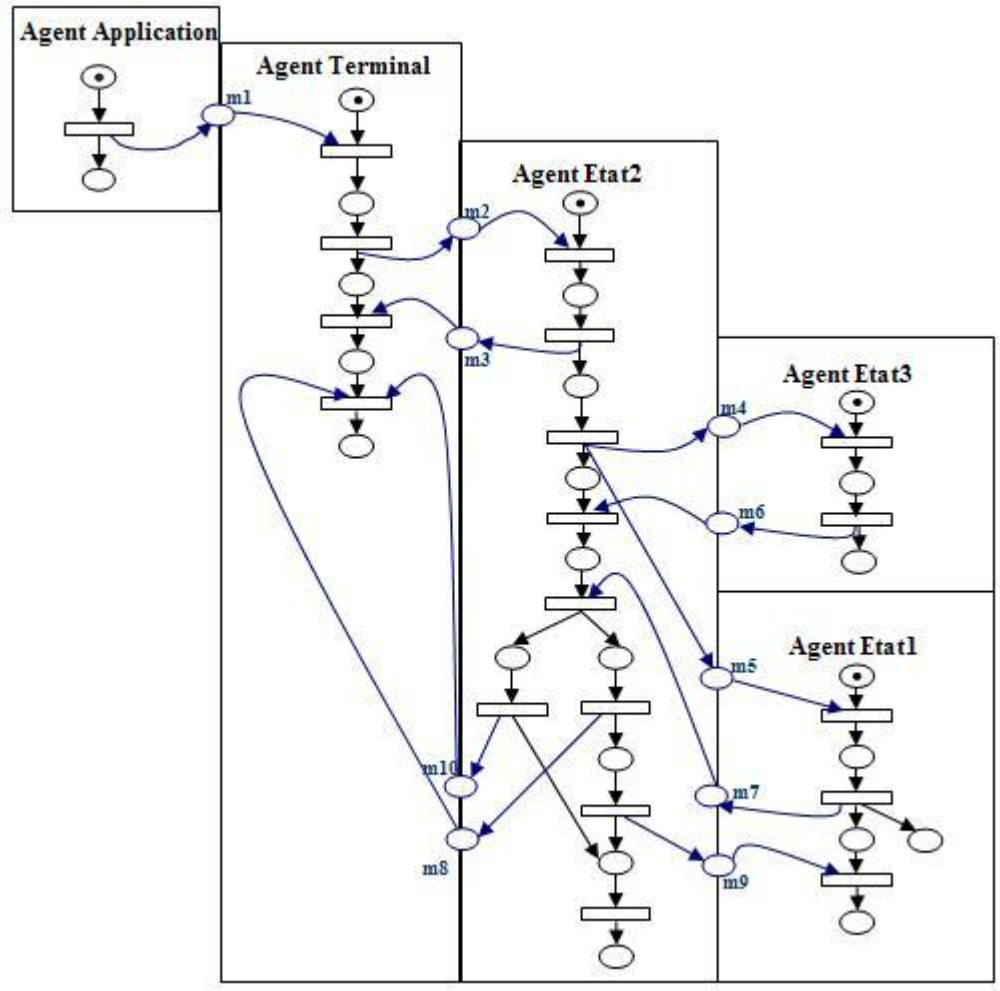


Figure 21. Modélisation des interactions entre les agents par un réseau de Pétri ordinaire.

Une simple vérification manuelle est suffisante pour vérifier le réseau de Pétri décrit précédemment, il faut lancer le jeton correspondant à l'agent Application et faire des franchissements d'étapes successives selon chaque événement dans le système. On peut vérifier ici que le modèle réseau de Pétri ne contient ni situation de blocage ni situation de conflit.

4. Références

[1] N. Samaan and A. Karmouch, "An Evidence-Based Mobility Prediction Agent Architecture", MATA 2003, pp. 230_239, Berlin.

[2] M. Jaseemuddin, An Architecture for Integrating UMTS and 802.11 WLAN Networks, Proceedings of IEEE Symposium on Computers and Communications (ISCC 2003), Antalya, Turkey, pp. 716-723, 2003.

[3] J. A. García-Macías, F. Rousseau, G. Berger-Sabbatel, L. Toumi, and A. Duda. «Différenciation des services sur les réseaux sans-fil 802.11». *Proc. Colloque francophone sur l'ingéniérie des protocoles*, Montreal, Canada,

[4] M. McGuire, K.N. Plataniotis and A.N. Venetsanopoulos. «Estimating position of mobile terminal from path loss measurements with survey data», *Wireless Communications & Mobile Computing*, vol. 3, pp. 51-62, February 2003.

5. Publications

[1] **B. Benmammar** and **F. Krief**. "MQoS NSLP : a mobility profile management based approach for advance resource reservation in a mobile environment". The 7th IFIP / IEEE International Conference on Mobile and Wireless Communications Networks (MWCN 2005). Marrakech, Morocco. September 19-21, 2005.

[2] **B. Benmammar** and **F. Krief**. "An Advanced Resource Reservation Protocol in Wireless Networks Based on User Mobility Profile". The Fifth IEEE Workshop on Applications and Services in Wireless Networks (ASWN 2005). Paris. June 29th – July 1st. FRANCE.

[3] N. Samaan, **B. Benmammar**, **F. Krief**, A. Karmouch. *"*Prediction-based Advanced Resource Reservation in a Mobile Environment*".* 18th IEEE Annual Canadian Conference on Electrical and Computer Engineering, (CCECE05), May 1-4, 2005, Saskatoon Inn, Saskatoon, Saskatchewan Canada.

[4] **B. Benmammar** and **F. Krief**. "Advance resource reservation in a WMAN environment based on the QoS NSLP signaling application and the CTP protocol". (MAN'05). IFIP Open Conference on Metropolitan Area Networks. Architecture, protocols, control, and management. HCMC, Viet Nam. April 11-13, 2005.

[5] **Z. Jrad**, **B. Benmammar**, **J. Corréa**, **F. Krief** and **N. Mbarek**. "A User Assistant for QoS Negotiation in a Dynamic Environment Using Agent Technology". Second IEEE and IFIP International Conference on Wireless and Optical Communications Networks (WOCN 2005). March 6-8, 2005, Dubai, United Arab Emirates UAE.

[6] **B. Benmammar** et **F. Krief**. "Gestion dynamique du handover horizontal et vertical basée sur le profil de mobilité de l'utilisateur". Colloque GRES 2005 : Gestion de REseaux et de Services, Du 28 Février au 3 Mars à LUCHON, France.

[7] **B. Benmammar**, "Les réseaux sans fil et la nouvelle signalisation IP". École DNAC d'hiver Sur le Nil, Égypte, du 11 au 18 décembre 2004.

[8] **B. Benmammar** et **F. Krief**. "La mobilité dans la future génération de protocoles de signalisation du monde IP". 6 ème Journées Doctorales Informatique et Réseau. (JDIR'04). Lannion, France Télécom R&D, 2-4 Novembre 2004.

[9] **B. Benmammar** and **F. Krief**. "Agents for Wireless Environments". International Conference on Telecommunication Systems, Modeling and Analysis. (ICTSM'2004). IFIP WG 7.3. Monterey, USA. Jully 2004.

[10, 11] En cours de soumission : Les Annales des Télécommunications, Net-Con 2005

Conclusion

Ce livrable complète le livrable 4.1.

La première partie de ce document à traité de la négociation dynamique de SLA/SLS à travers l'utilisation, d'une part, d'un assistant de négociation, placé dans le terminal utilisateur, qui permet de choisir le meilleur fournisseur de services et de négocier dynamiquement les paramètres de QoS et, d'autre part, la mise en œuvre d'un protocole de négociation de paramètres de SLS, conforme à l'environnement de signalisation NSIS, appelé SLN NSLP (Service Level Negotiation NSLP).

Concernant l'assistant de négociation, l'accent a été mis plus particulièrement sur la description du module d'apprentissage et l'implémentation de la plateforme multi-agents. De plus, l'assistant de négociation fera l'objet d'une démonstration lors de la revue finale du projet.

Concernant le protocole de négociation SLN NSLP, les interactions de ce protocole avec les autres protocoles de l'environnement NSIS ont été spécifiées et un cas d'utilisation pour la visioconférence ou encore la téléphonie sur IP avec SIP a été présenté.

La deuxième partie de ce livrable a traité de la mobilité du terminal. Deux approches permettant d'améliorer la qualité de service dans un réseau sans fil ont été présentées.

La première approche est basée sur la technologie agent qui est utilisée lorsqu'un nouvel utilisateur se présente, ce dernier est alors inconnu du système et la réservation de ressources à l'avance est donc impossible car ses futures localisations sont inconnues. L'approche agent vise durant cette phase à adapter le handover aux besoins de qualité de service de l'utilisateur. Les interactions entre les agents sont modélisées à l'aide de AUML et des réseaux de Pétri, c'est ce dernier outil qui nous a permis de faire une vérification du système.

La deuxième approche permettant d'améliorer la qualité de service dans un réseau sans fil, est une approche protocolaire qui est utilisée lorsque le profil de mobilité de l'utilisateur est connu par le système. Dans ce cas, les futures localisations de l'utilisateur sont déterminées et une réservation de ressources à l'avance, basée sur l'application de signalisation QoS NSLP, peut être faite pour lui. Un modèle mathématique est proposé et les résultats de la simulation à l'aide de MATLAB sont présentés.

Annexe : Les plateformes multi-agents

De nombreuses applications nécessitent l'élaboration de systèmes complexes distribués sur plusieurs machines qui interagissent, et qui doivent communiquer entre eux tout en s'exécutant indépendamment les uns des autres parfois même sous forme de sous-systèmes ou sous-tâches qui mènent à un but commun. Les méthodologies désormais classiques comme l'orienté objet, ne fournissent pas des concepts suffisants pour maîtriser toute la complexité des systèmes distribués, notamment pour modéliser des systèmes dans lesquels les rapports évoluent dynamiquement. Afin de pallier ces insuffisances, une évolution de la programmation orientée objet vers une programmation permettant de concevoir de façon naturelle ce type de systèmes s'est opérée depuis quelques années. Plusieurs méthodologies, qui constituent une extension des méthodologies orientées objet ou à base de connaissance, ont été proposées en particulier pour le développement des systèmes multi-agents (SMA). Il s'agit de systèmes constitués d'un ensemble d'entités autonomes appelés agents. Actuellement, de plus en plus d'applications adoptent les SMA, ce qui nécessite l'introduction de standards régissant la programmation et les méthodologies orientées agents, ainsi que l'élaboration de techniques de modélisation adéquates.

Par ailleurs, les SMA sont un des domaines de l'informatique où les logiciels sont les plus difficiles à développer, du fait du nombre important de préoccupations à prendre en compte : communication, gestion des ressources, recherche d'autres agents, etc. La notion de plateforme, liée à l'implémentation des systèmes multi-agents, constitue un endroit au sein duquel les agents peuvent évoluer. Les plateformes sont des environnements permettant de gérer le cycle de vie des agents et des services auxquels certains agents ont accès. En plus, peu d'efforts ont été fait ssur la standardisation des plateformes SMA. Il apparaît donc évident que le développement des SMA reste encore un domaine ouvert.

Le but de ce travail est de présenter certaines plateformes SMA que nous avons jugées intéressantes.

A.1 Vers une standardisation de la technologie multi-agents

Depuis plusieurs années des chercheurs ainsi que des industriels ont mené plusieurs travaux de recherche sur les agents. Cependant, peu d'efforts ont été consacrés à l'harmonisation des architectures SMA. Ce manque de consensus est dû en partie à l'absence d'une conception commune, dans le cercle de la recherche, des premiers principes généraux sur lesquels doivent se baser les architectures SMA [RIC 00]. Ce n'est que récemment que plusieurs groupes de chercheurs et d'industriels indépendants ont tenté de proposer une standardisation de la technologie multi-agents. Parmi ces groupes, on peut citer :

– Foundation for Intelligent Physical Agents (FIPA);

– Knowledgeable Agent-oriented System (KAOS);

– General Magic Group.

A.1.1. Le modèle de FIPA

FIPA[1] (Foundation for Intelligent Physical Agents) est un groupe multidisciplinaire qui poursuit la standardisation de la technologie agent. La mission que s'est fixée la FIPA est de faciliter l'interopérabilité des agents et des systèmes multi-agents provenant de différents fournisseurs [SAB 01]. Ainsi, la FIPA a produit depuis 1997 un ensemble de spécifications, qui s'étend des langages de communications (Agent Communication Languages) aux langages de contenu (Content Language), ainsi qu'aux protocoles d'interaction (Figure 1). Le langage FIPA-ACL suit le style de KQML (utilisant des performatifs issus de la théorie des actes de langage et quelques paramètres complémentaires), mais avec une sémantique mieux spécifiée. Le langage prévoit aussi l'utilisation de protocoles d'interaction.

[1] http:// www.fipa.org

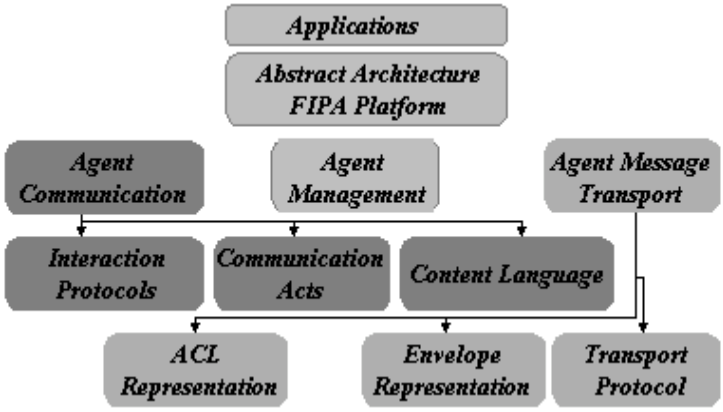


Figure 1. *Les spécifications de la FIPA*

FIPA spécifie les interfaces des différents composants de l'environnement avec lequel un agent peut interagir (c.à.d. humain, autres agents, autres logiciels) et le monde physique. Ainsi, la FIPA produit deux types de spécifications [BOI 01] :

– ''formatives'' pour décrire le comportement externe de l'agent et assurer l'interopérabilité avec les autres sous-systèmes spécifiés de la FIPA.

– ''informatives'' des applications représentant un guide pour l'industrie qui concerne l'utilisation de la technologie FIPA.

Pour la gestion des agents dans un environnement ouvert, l'approche de FIPA pour le développement des SMA est basée sur un paradigme décrit en utilisant, d'une part, un modèle référentiel qui spécifie un environnement normatif dans lequel les agents existent et opèrent, et d'autre part, une plateforme agent qui spécifie une infrastructure pour le déploiement et l'interaction des agents.

La plateforme proposée par la FIPA contient trois composants principaux qui sont évalués lors des tests d'interopérabilité : la couche de communication ou le *Message Transport System*, la gestion des agents ou le *Agent Management System* et la gestion de l'annuaire ou le *Directory Facilitator*. Le protocole utilisé est *Iiop* et la structure des messages est définie avec précision.

A.1.1.1Illustration

La Figure 2 détaille la structure d'un message FIPA ainsi que l'envoi d'un message entre deux agents situés sur des plateformes distinctes. L'échange de messages se passe entre deux agents A et B. L'agent A commence par construire le corps du message qui représente la sémantique de l'échange et l'enveloppe qui regroupe les informations de transport (encodage, protocole, ...). Il délègue ensuite l'expédition du message au *Message Transport System* qui, en fonction du protocole utilisé (Iiop, Http, Rmi, ...), sélectionne un *Message Transport Provider* et effectue la communication avec la plateforme hébergeant l'agent B.

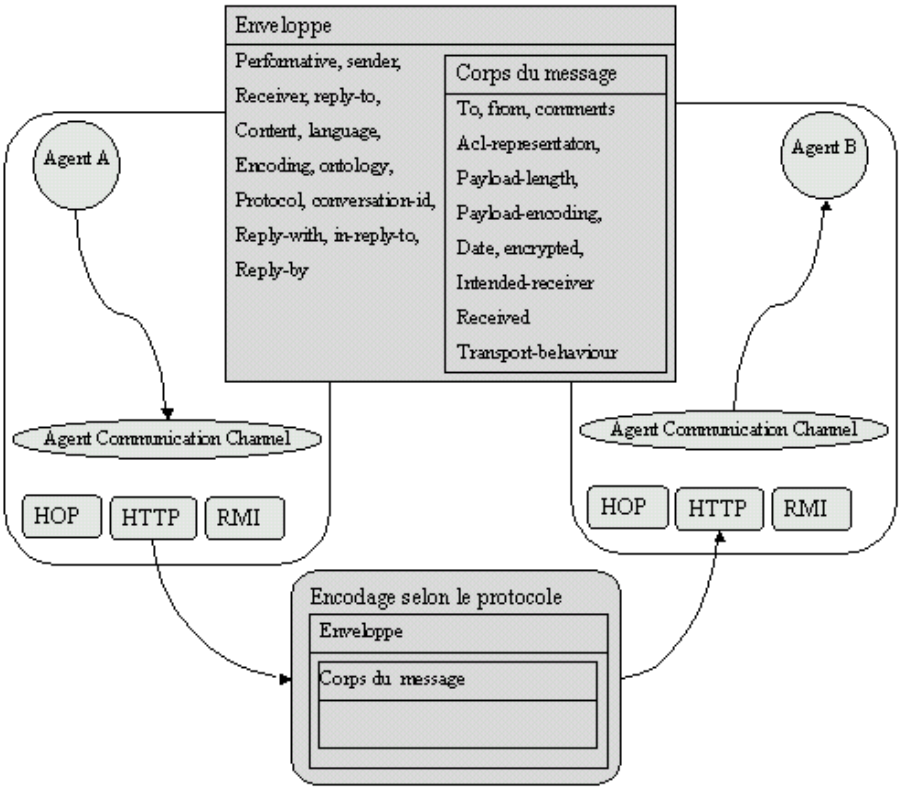


Figure 2. Echange de messages entre deux agents A et B de la FIPA

La notion de plateforme est utilisée comme une infrastructure de communication et de gestion des agents. Ces fonctionnalités sont apportées au travers de services de plateformes accessibles, soit aux agents de la plateforme, soit aux autres plateformes.

Le Tableau 1 présente des exemples de plateformes qui répondent aux spécifications de la FIPA.

Produit	Institut de recherche	Reference
Agent Development Kit	Tryllian BV	http://www.tryllian.com/
Zeus	Brittish Telecom	http://193.113.209.147/projects/ agents/zeus/
Fipa-Os	Emorphia	http://fipa-os.sourceforge.net/
April Agent Platform	Fujitsu	http://sf.us.agentcities.net/aap/
Grasshopper	IV++ Technologies AV	http://www.grasshopper.de/
JACK Intelligent Agents	Agent Oriented Software	http://www.agent-software.com/
Java Agent Development Environment (Jade)	TILAB (CSELT)	http://sharon.cselt.it/projects/ jade/
Lightweight Extensible Agent Platform (Leap)	Motorola	http://leap.crm-paris.com/

Tableau 1. *Exemples de plateformes répondant aux spécifications de la FIPA.*

A.1.2. Le modèle de KAOS

Le groupe KAOS (Knowledgeable Agent-oriented System) propose quant à lui une architecture SMA distribuée et ouverte pour les agents logiciels. KAOS [BRA 96] a comme objectif principal de traiter les deux limitations majeures de la technologie agent. Le premier problème est traité en profitant des capacités des produits commerciaux d'objet distribué (CORBA, DCOM, Java), comme fondation pour la fonctionnalité agent et en

supportant la recherche et les efforts de standardisation pour résoudre les problèmes d'interopérabilité des agents. Le deuxième problème est traité en apportant une méta-architecture d'agent de communication dans laquelle tout langage de communication accompagné par sa sémantique peut être accommodé. A l'opposé de la plupart des architectures d'agents de communication, KAOS prend explicitement en considération non seulement le message individuel mais aussi les séquences variées des messages dans lesquels cela peut arriver. L'architecture décrit les implémentations des agents (allant de la notion de simple agent à la notion de rôle d'agent, tel que les médiateurs) et élabore, sur la base des interactions dynamiques d'agent à agent, des messages de communication en utilisant des politiques de conversation.

A.1.3. Le modèle de General Magic

General Magic est une tentative de recherche commerciale sur la technologie d'agents mobiles pour le commerce électronique. Le langage Telescript de General Magic [JIM 96] est un langage riche orienté objet qui soustrait aux programmeurs de nombreux aspects de la mobilité, de la sécurité, de la durée des objets et de l'interaction des processus. Il est basé sur des agents mobiles qui se déplacent de serveur en serveur (''places'' Telescript) et peuvent interagir localement avec les applications hébergées par le serveur. Cette infrastructure permet de développer des applications de commerce électronique où un agent peut effectuer des requêtes sophistiquées pour le compte d'un utilisateur, en se déplaçant de place en place et en interagissant avec les applications (médiateurs, offreurs de services…) qu'elles hébergent, suivant le schéma contenu dans son code Telescript.

Conceptuellement, cette technologie modélise un SMA comme un marché électronique où les fournisseurs et les consommateurs peuvent se rencontrer et faire des transactions d'affaires. Les agents mobiles sont représentés comme des entités qui résident dans un endroit particulier à un instant donné et ce marché est modélisé comme un réseau d'ordinateurs qui supporte une collection de lieux qui offrent des services aux agents mobiles. Les compétences des agents sont décrites dans les points suivants :

– ils peuvent se déplacer d'une place à une autre et ont le droit de voyager ;

– ils peuvent rencontrer et invoquer des procédures d'autres agents ;

– ils peuvent créer des connexions pour communiquer avec un agent d'une place différente ;

– ils possèdent une représentativité qui indique le monde physique individuel ou l'organisation que l'agent représente ;

– ils ont des permis qui indiquent les compétences des agents ;

A.2. Caractéristiques d'une plateforme multi-agents

Les capacités de l'agent à être autonome, en interaction et à se situer dans un environnement, sont parmi les caractéristiques fondamentales connues des agents. Mais ces caractéristiques ne permettent pas de définir tous les types d'agents possibles, on prendra par exemple les agents de type B.D.I. qui impliquent l'ajout d'un certain nombre de caractéristiques *cognitives* aux agents, ou les agents communiquant par des langages tels que KQML qui nécessitent des fonctionnalités supplémentaires.

Une plateforme n'a donc pas pour vocation de proposer tous les types d'agents possibles, mais seulement un ensemble de fonctionnalités. La Figure 3 montre comment le développeur peut combiner ces fonctionnalités à volonté pour engendrer plusieurs types d'agents possibles.

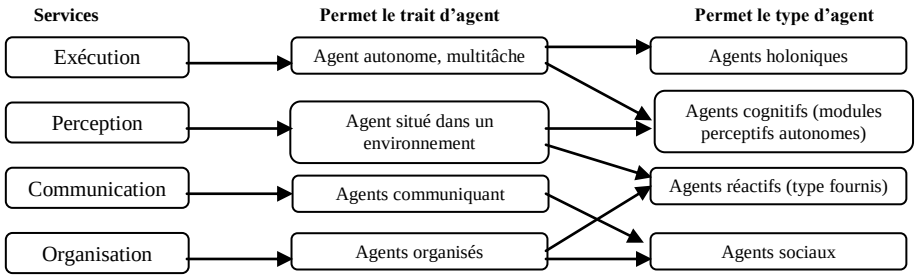


Figure 3. *Types d'agents composés à partir de services*

Pour permettre de réaliser un large ensemble d'agents, Pierre [PIE 03] pose l'hypothèse selon laquelle il suffit de définir un ensemble de fonctionnalités de base et un mécanisme d'extension. Ainsi, il définit les exigences nécessaires pour construire une plateforme permettant de réaliser des systèmes multi-agents.

A.2.1. Exigences méthodologiques pour une plateforme de simulation multi-agents

Dans le but de participer à la construction d'une structure générale qui caractérise un type idéal de plateforme[2], [SIM 01] vise trois objectifs différents. Le premier est de définir un modèle de référence pour une spécification des besoins d'une plateforme de simulation multi-agents. Le second est d'établir une analyse comparative entre les différentes plateformes disponibles pour le comité de recherche. Le troisième objectif est la spécification, le design et l'implémentation d'une plateforme qui suivent une partie de ces besoins et qui se focalisent sur la simulation des agents. Nuno et al [NUN 02] rapportent leurs résultats par rapport au deuxième et troisième objectif. Et Marietto et al [MAR 02] traite le deuxième objectif et met en évidence des exigences pour une plateforme de simulation agent. Ces exigences s'articulent autour des axes suivants :

A.2.1.1. Axe technique

Selon cet axe, une plateforme de simulation doit offrir des services qui permettent l'utilisation des systèmes d'exploitation et les ressources associées, notamment le réseau. Elle doit également offrir des services pour l'exécution contrôlée et reproductible de simulations.

A.2.1.2. Axe de domaine

La plateforme doit être capable de proposer un ensemble de modèles d'agents pour guider la conception des agents. Elle doit également offrir une gestion des erreurs (comme les messages corrompus) au niveau technique et comportemental de la simulation.

A.2.1.3. Axe de développement

La plateforme doit permettre de gérer les méthodes de communication et l'utilisation de différents langages de communication agent. Elle doit également mettre à disposition des modèles d'architectures d'agents, des agents réactifs et cognitifs. Enfin, l'un des aspects les plus importants est du fait que la plateforme doit offrir des abstractions pour l'organisation des agents et doit pouvoir gérer plusieurs ensembles d'agents.

Les deux autres axes qui ont été développés dans [MAR 02] sont l'analyse et l'exploration qui impliquent que les agents soient cognitifs, et qui sont par ailleurs trop spécifiques aux besoins des *simulations* à base d'agent utilisées dans le domaine scientifique. Malgré ces divergences, les trois axes (technique, domaine et développement) semblent être tout à fait pertinents pour la conception d'une plateforme généraliste. On peut aussi remarquer qu'une plateforme peut avoir une influence sur la conception du système à base d'agent, par exemple la plateforme MadKit [FER 89] offre son méta-modèle organisationnel AALADIN, mais que ce n'est pas une exigence. Pour éviter de restreindre le développeur, il est même préférable que la plateforme ne propose pas de méthodologie ou de modèle trop affirmé. Il ne faut pas non plus laisser le développeur à l'abandon : la plateforme doit également guider le développeur dans l'analyse et la conception de son système.

A.2.2. Autres formes d'exigences pour une plateforme agent

Les exigences proposées ci-dessus sont basées sur des aspects méthodologiques (technique, domaine, développement, analyse, etc.). L'inconvénient de cette orientation méthodologique est qu'il est difficile de représenter les différentes entités nécessaires à la réalisation d'un système agent. Ainsi, Pierre [PIE 03] propose une classification des exigences d'une plateforme agent en fonction de différentes entités:

– l*es agents* qui sont les composants du système à créer,

– l'*environnement* qui représente les ressources du système à créer, ce qui inclue également les agents,

– l*a plateforme* qui est le logiciel qui permet aux agents d'évoluer dans l'environnement, de manière autonome et d'interagir entre eux.

[2] http://www.lti.pcs.usp.br/SimCog.

Une définition plus précise des rôles de chaque élément de la proposition de l'architecture d'une plateforme agent ci-dessus est nécessaire pour compléter la classification des exigences énoncées précédemment. Ainsi, une liste des exigences pour une plateforme agent peut être définie de la manière suivante:

– agents :
  - réalisation de la caractéristique d'autonomie
  - réalisation de la caractéristique d'interaction
  - offrir un ensemble de types d'agents prédéfinis
  - permettre la construction de types d'agents particuliers

– environnement :
  - permettre aux agents de s'organiser
  - permettre aux agents d'accéder aux ressources du système
  - offrir un accès transparent aux ressources quelque soit leur localisation physique

– plateforme :
  - offrir une infrastructure pour faire vivre les agents
  - offrir une infrastructure pour gérer l'environnement
  - permettre de spécialiser les fonctionnalités existantes ou d'en ajouter de nouvelles.

On pourrait également enrichir ces exigences avec les notions de *sécurité* et de *migration*. En effet, si l'on veut pouvoir créer des agents mobiles, il faut que la plateforme mette à disposition un mécanisme de sécurité, permettant de restreindre l'accès aux ressources de l'environnement. Ce service de sécurité reposerait sur un mécanisme de protection géré au niveau de l'environnement. La migration d'agents serait également fournie par un service permettant la persistance des agents (car la migration implique l'utilisation de la persistance), qui doit également être prévue dans la plateforme.

A.3. Evaluation des plateformes multi-agents

L'élaboration de critères d'évaluation des plateformes multi-agents a fait l'objet de nombreux travaux de recherche. Garneau [GAR 02], par exemple, a choisi un barème allant de 0 à 4 et a donné ainsi une note reflétant le niveau de la plateforme selon les critères d'évaluation suivants :

– la méthodologie liée à l'outil couvre les différentes étapes ;

– facilité d'apprentissage de l'outil ;

– facilité de transition entre le développement et l'implémentation ;

– souplesse de l'outil ;

– communication inter-agents ;

– outils de "débuggage" ;

– support de développement ;

– support pour la gestion du SMA ;

– diminution de l'effort demandé et simplicité d'implémentation ;

– support pour les bases de données ;

– génération automatique de code ;

– extensibilité du code ;

– déploiement ;

– documentation.

Un autre travail d'évaluation est celui de Ricordel et al [RIC 00] qui considère les systèmes multi-agents comme des logiciels complexes qui nécessitent des méthodes adéquates telles que les quatre étapes de construction : l'analyse, la conception, le développement et le déploiement. Ces quatre étapes sont définies de la manière suivante :

– analyse : le processus de détermination, de séparation et de description du type de problème et de son domaine. En pratique, il consiste à identifier le domaine d'application et la clé du problème ;

– conception : le processus de définition de l'architecture solution du problème. Il consiste à spécifier une solution principale du problème, par exemple, en utilisant UML ;

– développement : le processus de construction d'une solution fonctionnelle du problème. En pratique, il consiste à coder la solution avec un langage de programmation particulier ;

– déploiement : le processus d'application de la solution à un problème réel.

Il a ensuite déterminé quatre mesures de qualités des étapes précédentes qui touchent à plusieurs aspects du développement pratique d'un système SMA et qui sont :

– la complétude : elle spécifie la quantité et la qualité de la documentation et des outils fournis ;

– l'applicabilité : elle précise les possibilités offertes et les restrictions imposées ;

– la complexité : elle inclut la compétence demandée des développeurs et la quantité de travail nécessaire pour accomplir la tâche ;

– la réutilisation : la quantité de travail gagnée en réutilisant un travail ancien.

Ainsi, ces mesures de qualité appliquées aux étapes de développement mentionnées ci-dessus aboutissent à plusieurs critères d'évaluation d'une plateforme tel que :

– complétude de l'analyse de la plateforme,

– applicabilité de l'analyse de la plateforme,

– complexité de l'analyse de la plateforme,

– réutilisation de l'analyse de la plateforme,

– complétude de la conception de la plateforme,

– etc.

A.4. Exemples de plateformes SMA

On a présenté ci-dessus des travaux, soit de standardisation, soit de simple définition des exigences des plateformes agent. Cependant, toutes les plateformes ne répondent pas forcément à ces critères. La plupart d'entre elles sont conçues, soit par rapport à un modèle d'agent particulier comme *Zeus,* soit par rapport à un domaine d'application particulier comme les agents mobiles *Aglets,* ou la simulation *Swarm.*

En effet, il existe un grand nombre de plateformes multi-agents dédiées à différents modèles d'agents. Nous ne prétendons pas donner ici une liste exhaustive de l'ensemble des plateformes disponibles actuellement. Nous donnons une description succincte de quelques-unes choisies selon trois angles de vues différents :

– Plateformes pour simulation telles que Swarm, Cormas, StarLogo, Geamas, Mice…

– Plateformes pour implémentation comme Zeus, AgentBuilder, JACK, MadKit, DECAF, MAGIQUE, MACE, Mask, MoCAH, OSACA …

– Plateformes pour mobilité : Aglets, Concordia, Gossip, Grasshopper, Gypsy, JumpingBeams, GenA, KlAIM, Mole …

A.4.1. Plateformes pour simulation

Une simulation permet de construire une abstraction de la réalité (un modèle) et de faire évoluer cette abstraction en fonction du temps. L'objectif d'un modèle de simulation est de reproduire les activités des différentes entités du système simulé et donc d'apprendre quelque chose sur les comportements ou les performances de ce système. Il existe un certain nombre de plateformes qui aident à former une simulation du système multi-agents pour prédire ainsi le comportement de ses agents. On présente dans le Tableau 2 quelques plateformes reconnues pour faire de la simulation et on discute de quelque-unes dans la suite.

Produit	**Institution de recherche**	**Langage**	**Caractéristiques**
CORMAS	Cirad	SmallTalk	- Plateforme pour faire de la simulation – Pour les problèmes de dynamique et d'usage de ressources – Plateforme téléchargeable, évolutive – Plusieurs applications sur la simulation de

			sociétés d'agents exploitant une ressource pouvant être dynamique et simulée par un SMA – L'espace est un automate cellulaire – 2 types d'agents : spatialisés et communicants – Maximum de 5000 agents réactifs
GEAMAS	Mas - La Réunion	Java	– Pour modélisation et simulation – Plateforme évolutive en java – Plusieurs applications sur la modélisation des phénomènes sismiques et volcaniques, gestion de déchets, étude de la ressource pélagique dans l'environnement océanique – Micro noyau générique auquel peuvent être couplées des extensions (apprentissage, auto-organisation, …) – Environnement tient la place centrale, seul moyen de communication – agents de réactifs à 'fortement cognitifs'
SWARM	SWARM Developme-nt Group	Objective C, Java	- Plateforme pour Simulation - Possibilités de hiérarchies et de structures récursives. - De nombreuses applications développées, notamment à caractère écologique.
StarLogo	Media Laboratory, MIT, Cambridge	StarLogo	Un environnement de programmation pour explorer les comportements de systèmes décentralisés

Tableau 2. *Des plateformes pour la simulation*

A.4.1.1. CORMAS

Créé par l'équipe GREEN (Gestion des ressources renouvelables, environnement) au sein du centre de recherche CIRAD[3] (centre de recherche en agronomie pour le développement, Monpellier), CORMAS (Common-Pool Resources and Multi-Agent System) est un environnement de développement de SMA dédié aux simulations et à la résolution de problèmes de dynamique et d'usage de ressources. Conçu comme une couche supplémentaire au-dessus de l'environnement de programmation SmallTalk *VisualWorks*, CORMAS fournit des bibliothèques d'agents, de modèles et de protocoles qui permettent à l'utilisateur de se construire son modèle (en s'appuyant éventuellement sur des modèles préexistants) pour le traiter par simulation. Selon les besoins de l'application, CORMAS permet de disposer d'agents très réactifs ou délibératifs spatialisés (situés dans l'espace), communicants, socialement contrôlés ou non, raisonnant sur le temps ou non, stratèges ou opérationnels.

La conception de modèles sous CORMAS consiste en trois étapes principales :

– la définition des agents, de l'espace et des communications ;

– la définition de l'initialisation et du contrôle de la simulation ;

– la définition de l'observation de cette simulation.

Les agents définis, à partir de classes, lors de la première phase peuvent être de plusieurs types (Agent, AgentSitué, AgentCommuniquant, AgentSituéCommuniquant). Il s'agit de définir des entités actives ou passives pour la simulation. Les entités passives peuvent être des objets de la simulation (qu'ils fassent partis de l'environnement ou non) ou des messages que s'échangent les entités communicantes de la simulation.

Pour compléter la première phase, il faut que l'utilisateur définisse les communications ou plus précisément les messages et les protocoles de communication entre les agents communiquant. L'utilisateur a également la possibilité de visualiser un graphe des communications entre acteurs semblable au paradigme GraphTracer fourni avec le JDK 1.2.

La définition de l'espace est facultative et elle correspond à la construction de l'automate cellulaire qui représente la dynamique spatiale du modèle, c'est à dire la définition des cellules et de leur dynamique. La

[3] http://www.cirad.fr/presentation/programmes/espace/cormas/eng/ index.shtml

dynamique des cellules de l'automate cellulaire est, pour plusieurs modèles, définie sans prendre en compte les effets de voisinage très emblématiques des automates cellulaires.

Lors de la deuxième phase, il faut initialiser les différents agents et objets en instançiant les variables internes. Il faut également définir le protocole de contrôle de l'évolution de la simulation. Malheureusement, CORMAS ne dispose pas d'un scheduler d'agents pré-implémenté, il faut en fait implémenter une méthode qui correspond à un tour ou une période de la simulation. Pourtant, il n'est pas facile de s'abstraire au niveau de la conception des agents en supposant acquit (du point de vue implémentation), le fait que les agents s'exécutent en parallèle comme cela peut être le cas sur d'autres platesformes où l'on peut faire abstraction du scheduler.

La dernière étape consiste à définir l'observation de la simulation. Premièrement, il faut déterminer combien de fois ou combien de temps va durer la simulation. Deuxièmement, il faudra déterminer, quand on a une interface graphique, les paramètres généraux de cet espace, la dimension de la grille (en nombre de cellules, le type de voisinage des cellules (4-connexe, 6-connexe, 8-connexe), si l'espace est toroïdal (espace fermé) ou non. Pour cela, on détermine si l'exécution va être réalisée pas à pas ou en continu, et, si la simulation est réalisée en continu combien de périodes vont être réalisées. Ce qui correspond au nombre de fois où la méthode d'évolution de la simulation sera appelée.

Il est tout à fait possible, si l'interface d'observation graphique ne convient pas à l'observation du phénomène du modèle, d'en réaliser une autre sous VisualWorks, comme des courbes d'évolution de la démographie par exemple.

A.4.1.2. SWARM

Swarm est une plateforme générique, constituée d'une bibliothèque logicielle (en Objective C, avec une interface Java) qui permet de développer des simulations à base d'agents. Sa conception originale est due à Langton[4] (artificial life) et s'est fait à l'Institut Santa Fe (Nouveau Mexique). L'objectif étant de définir une plateforme de développement permettant de faire des "expérimentations informatiques" décentralisées à évènements discrets de systèmes complexes. C'est un environnement de simulation de sociétés d'agents à grande échelle (en nombre d'agents).

Dans Swarm une simulation est constituée d'un ensemble d'agents et d'un calendrier indiquant ce qu'un agent peut envoyer comme message et à quel moment. L'association des agents au calendrier définissant la dynamique des échanges constitue un modèle dans Swarm. Cette plateforme est très reconnue dans le domaine de la vie artificielle.

Les démarches à suivre pour la conception de modèles sous SWARM :

– création du monde virtuel : environnement artificiel spatial et temporel où évoluent des entités ;

– création d'agents d'observation qui sondent le monde vituel précédent ;

– fonctionnement de l'ensemble (monde virtuel + observateurs) dans une simulation discrétisée en temps avec des horloges synchronisées.

A.4.1.3. GEAMAS

GEAMAS (GEneric Architecture for Multi-Agents Simulations) est une plateforme logicielle générique pour la modélisation et la simulation multi-agents entièrement implémentée en JAVA. Elle a été développée à l'Université de la Réunion par l'équipe MAS2. L'architecture logicielle de GEAMAS est structurée en trois modules : un micro-noyau, un environnement de génération et un environnement de simulation. Le micro-noyau générique, JAAFAAR [CAL 99], offre les structures et les mécanismes minimaux nécessaires à l'implémentation du SMA, autour duquel gravitent un certain nombre d'extensions logicielles spécialisées (modules d'apprentissage, d'auto-organisation, de conception assistée, etc.). L'environnement de génération permet le design graphique des applications et l'environnement de simulation permet l'observation de l'évolution de la simulation à travers l'outil d'interface graphique de l'utilisateur.

[4] Swarm du Santa Fe Institute : http://www.swarm.org/

A.4.2. Plateformes pour implémentation

Ces plateformes servent à implémenter des architectures d'agents qui sont conçues dans le but de faciliter le développement des agents. Les architectures existantes varient des simples objets actifs (par exemple : Actalk), ou des objets actifs dotés d'un langage de communication tels que KQML, à des architectures plus complexes basées sur des propositions conceptuelles telles que les états mentaux.

Les plateformes pour implémentation peuvent être réparties en sous-groupes tels que conception et implémentation, ou conception, implémentation et validation comme c'est le cas dans le travail de [GUE 01]. On se restreint ici à la présentation de quelques unes (Tableau 3) sans rentrer dans les détails de la de la conception et de la validation.

Produit	**Institution de recherche**	**Langage**	**Description**
Madkit	Madkit Developme nt Group	Java Scheme, Jess	Multiagent Development Tool – Agents programmés –Notions de rôles, groupes, communautés – Messages = informations – Moteur synchrone – Interface – Organisation libre
Magique		Java	– Agents vides – Ajout de compétences – Échanges de compétences – Messages = actions – Utilisable sans interface – Interface optionnelle – Structure hiérarchique
OSACA	UTC Compiègne		- Open System for Asynchronous Cognitive Agents, - Réalisation de SMA avec peu d'agents cognitifs complexes
MASK	MAGMA (LEIBNIZ), Grenoble	Librairie Tcl/Tk, C, C++, Java	– MultiAgent System Kernel – Fournir des bibliothèques d'agents (A), de manipulation d'environnement (E), d'interaction (I) et d'organisation (O) ainsi que les outils d'aide à la programmation. – Agents cognitifs
MERCURE	LIMSI, AEGIS, LGIS	SmallTalk	– Conforme à la FIPA – Développement de SMA – Commercial – agents techniques spécifiés par la FIPA (Directory Facilitator, Agent Management Service, Agent Communication Channel, Agent Naming Service) et agents spécialisés. – Communications par un ACL – agents cognitifs
DIMA	LIP6 (Paris 6), LERI (Reims)	Basé sur SmallTalk puis Java	– Développement et Implémentation de Systèmes Multi-Agents – Pour simulations, résolutions de problèmes, systèmes de contrôles avec contraintes de temps réels – Plateforme évolutive – Plusieurs applications sur la ventilation artificielle, simulation de modèles économiques – Découpe un agent en module de perception, de délibération et de communication – Notion de méta-comportement – 3 types d'agents : réactifs, cognitifs, hybrides
Zeus	British Telecommu	Java	*Agent Building Environment*

	nications Labs		
Agent Builder	Reticular Systems, Inc.	Java	- Integrated Agent and Agency development - Environment composée de deux modules : - Toolkit pour définir les agents, les agences et les communications - Run-Time system qui est un interpréteur du système.
JACK Intelligent Agents	Agent Oriented Software Pty. Ltd.	Jack agent Language	- *Agent development environment*
dMars	Australian Artificial Intelligence Institute Ltd.	C, C++	*Agent-Oriented Development and Implementation Environment*
Jade	Tilab	Java	Environnement pour implémentation

Tableau 3. *Plateformes pour implémentation*

A.4.2.1. Magique

Magique[5] est une plateforme générique de développement de SMA, développée en JAVA (JDK 1.x) et destinée aux résolutions de problèmes distribués. Magique est une architecture distribuée sur réseau hétérogène. A la base, c'est un modèle d'organisation d'agents qui propose une organisation hiérarchique et qui utilise des agents réactifs et pro-actifs.

La structure de Magique permet de proposer un mécanisme de délégation de compétences entre agents qui peut faciliter le développement de systèmes distribués. Magique existe sous la forme d'une API java permettant le développement de systèmes multi-agents hiérarchiques constitués d'agents. Le concepteur de SMA peut s'appuyer sur les fonctionnalités offertes par Magique telles que la communication entre agents, la distribution de l'application, l'évolution dynamique, etc., pour développer les compétences applicatives nécessaires [MAT 01].

Parmi les diverses applications multi-agents[6] qui peuvent être réalisées en utilisant Magique, on peut citer :

– un système de vente aux enchères avec des agents acheteurs à stratégies différentes et un agent commissaire-priseur,

– le parcours d'un territoire par des agents explorateurs,

– des jeux multi-joueurs avec un agent arbitre-ordonnanceur et des agents joueurs,

– etc.

Les agents dans Magique

Dans Magique, il s'agit de répartir les compétences parmi les agents, puis d'organiser ces agents dans une hiérarchie [SEC 03]. Un agent est une entité qui possède un certain nombre de compétences qui lui permettent de tenir un rôle dans une application multi-agents. Grâce aux échanges avec les autres agents, les compétences d'un agent peuvent évoluer dynamiquement au cours de l'existence de celui-ci, ce qui implique que les rôles qu'il peut jouer et son statut peuvent évoluer au sein du SMA. Un agent est construit dynamiquement à partir d'un agent élémentaire "vide", par enrichissement ou acquisition de ses compétences. Dans l'implémentation, une compétence peut être perçue comme un composant logiciel regroupant un ensemble cohérent de fonctionnalités et les compétences peuvent être développées indépendamment de tout agent et donc réutilisées dans différents contextes. L'ajout de nouvelles compétences dans une application SMA est facile. Le cycle de vie d'un agent dans Magique consiste à exécuter l'algorithme suivant :

– à la réception d'une requête, exprimée sous la forme d'un nom de méthode et d'une liste de paramètres, l'agent vérifie si sa classe définit la méthode demandée,

[5] http://www.lifl.fr/SMAC/projets/magique/

[6] Différents petits exemples http://www.lifl.fr/MAGIQUE rubrique Toy Problems.

– si l'agent possède cette méthode, il l'invoque et retourne le résultat à l'agent ayant émis la requête,

sinon, il recherche dans son équipe si l'un de ses subalternes peut répondre à la requête,

– si c'était le cas, il lui demande de traiter la requête, sinon il délégue à son supérieur hiérarchique la gestion de cette requête.

Le modèle organisationnel : Magique

Le développement d'un système multi-agents avec Magique consiste à définir dans un premier temps un ensemble de classes Java caractérisant les différentes classes d'agent. Dans un deuxième temps, l'organisation de ces agents était définie au travers d'une hiérarchie. Ensuite, les agents surchargeant leur méthode *action()* initiaient les interactions au sein du système. Cette approche du développement d'un système multi-agents est très proche du développement de logiciel classique : après une phase d'analyse, le système est implémenté et ensuite testé.

L'organisation des agents peut être amenée à évoluer dynamiquement si une réorganisation de la structure du SMA se justifie en cours d'utilisation. Dans Magique, cette dynamique opère à plusieurs niveaux :

– individuel : un agent peut acquérir ou oublier des compétences suite à un échange avec les autres agents.

– relationnel : des liens d'accointance peuvent être crées lorsque des relations favorisées apparaissent entre deux agents de la hiérarchie.

– organisationnel/architectural : des agents peuvent être créés ou détruits afin d'adapter le système à certaines contraintes. Deux exemples parmi d'autres:

- un agent peut se voir submerger par un afflux de requêtes pour l'une de ses compétences ; il peut alors décider de créer une équipe d'agents qui auront cette compétence vers laquelle il pourra rediriger tout ou partie des requêtes qui lui parviennent,

- un agent doit pouvoir quitter temporairement le système et retrouver sa place ultérieurement, les messages qui lui sont destinés doivent alors avoir été mis en attente.

A.4.2.2. Zeus

Zeus, conçu par British Télécom, est un environnement complet, fondé sur les travaux de la FIPA, qui utilise une méthodologie appelée « role modeling », pour développer des applications collaboratives. Ses concepts s'appuient sur les notions d'agents, de buts, de tâches (que doivent réaliser l'agent pour atteindre son but) et de faits (ce que l'agent considère comme vrai). Zeus[7] fournit un environnement de développement d'agents grâce à un ensemble de bibliothèques Java que les développeurs peuvent réutiliser pour créer leurs agents. Zeus propose aussi un ensemble d'agents utilitaires (serveur de nommage et de facilitation) pour faciliter la recherche d'agents.

Zeus contient un ensemble de composants, écrits en langage de programmation Java, qui peuvent être regroupés en trois catégories de fonctionnement (ou Librairies): *Agent Component Library*, *Agent Building Tool*, *Agent Utilities* qui contient un *nameserver*, un facilitateur et un visualiseur d'agent. L'outil est l'un des plus complets. Les différentes étapes du développement se font à l'intérieur de plusieurs éditeurs: ontologie, description des tâches, organisation, définition des agents, coordination, faits et variables ainsi que les contraintes.

Les agents de Zeus

L'architecture des agents de Zeus est similaire à la majorité des architectures d'agents collaboratifs. Les agents ont les caractéristiques suivantes :

– un agent possède une *définition* ;

– il appartient à une *organisation* ;

– il *perçoit son e*nvironnement ;

– il *modifie* son environnement ;

– il possède un *cycle* de vie ;

– il s'appuie sur un *protocole* d'interaction.

[7] Zeus Agent Building Toolkit : http://www.labs.bt.com/projects/agents/zeus/index.htm

Un agent dans Zeus est constitué de trois couches :

– la couche de définition, qui contient les capacités de raisonnement et des algorithmes d'apprentissage. L'agent est vu comme une entité autonome capable de raisonner en termes de ses croyances, de ses ressources et de ses préférences.

– la couche organisationnelle, qui contient la base de connaissances de l'agent, ainsi que les relations entre les agents.

– la couche de coordination où l'on décide des modes de communication entre les agents, des protocoles, de la coordination et des autres mécanismes d'interactions.

L'architecture de Zeus se prête bien aux applications de planification ou d'ordonnancement, ce qui la rend particulièrement intéressante pour la réalisation de simulation ou d'applications de supervision. Cependant, chaque agent fonctionne grâce à une machine virtuelle Java (Jvm, Java Virtual Machine), ce qui devient rapidement difficile à gérer lorsque le nombre d'agents dans le système devient important.

A.4.2.3. Jade

Jade[8] (pour Java Agent DEvelopment framework) est un outil qui répond aux normes de la FlPA. C'est un middleware écrit en Java et se conformant aux spécifications de la FIPA. Cet environnement simplifie le développement d'agent en fournissant les services de base définit par la FIPA, ainsi qu'un ensemble d'outils pour le déploiement. L'outil possède trois modules principaux (nécessaires aux normes PIPA) : le DF « director facilitor » fourni un service de pages jaunes à la plateforme ; le ACC « agent communication chanel » gère la communication entre les agents ; le AMS « agent management system » supervise l'enregistrement des agents, leur authentification, leur accès et utilisation du système. Les agents communiquent par le langage FIPA ACL. Un éditeur est disponible pour l'enregistrement et la gestion des agents. Aucune autre interface n'est disponible pour le développement ou l'implémentation. À cause de cette lacune, l'implémentation demande beaucoup d'efforts. Elle nécessite une bonne connaissance des classes et des différents services offerts.

Une plateforme Jade peut être répartie sur un ensemble de machines et configurée à distance. Cependant, la configuration du système peut être altérée dynamiquement, à cause d'un mécanisme de migration d'agents au sein de la même plateforme.

A.4.2.4. AgentBuilder

La plateforme AgentBuilder[9] est un environnement de développement complet. Une modélisation orientée objet constitue la base de la conception des systèmes à laquelle on ajoute une partie « ontologie ». C'est une implémentation proche du langage Agent-0. Les agents sont décrits avec le langage Radl, Reticular *Agent Definition Language*, qui permet de définir les règles du comportement de l'agent. KQML est utilisé comme langage de communication entre les agents et les règles se déclenchent en fonction de certaines conditions et sont associées à des actions. Les conditions portent sur les messages reçus par l'agent tandis que les actions correspondent à l'invocation de méthodes Java. L'exécution du système se fait à partir de l'engin d'exécution d'AgentBuilder. Par contre, on peut créer des fichiers « .class » et les exécuter sur une JVM (*Java Virtuel Machine*) standard. Il est aussi possible de décrire des protocoles définissant les messages acceptés et émis par l'agent. AgentBuilder est probablement la plateforme commerciale la plus aboutie. AgentBuilder est un outil complexe qui demande des efforts d'apprentissage importants et de bonnes connaissances dans le domaine des systèmes multiagents pour être utilisé de façon performante. Il est limité au niveau de l'extensibilité, du déploiement et de la réutilisation.

A.4.2.5. Jack Intelligent Agents

Jack Intelligent Agents[10] est un environnement constitué d'un éditeur gestionnaire de projet, d'un langage de programmation JAL (Jack Agent Language), qui est une extension à Java, et d'un compilateur. Il est destiné au

[8] http://sharon.cselt.it/projects/jade/
[9] http://www.agentbuilder.com/
[10] http://www.agent-software.com.au/

développement d'agents de type BDI, tout en fournissant une API orientée objets. Le langage proposé au développeur est proche de la programmation logique et peut être ensuite compilé pour le transformer en classes Java. Les classes générées étendent des classes fournies par l'API du noyau de Jack : la classe Agent, la classe Event ou encore la classe Plan. Le noyau de Jack gère aussi la concurrence des tâches entre les agents, la réaction aux évènements et l'infrastructure de communication. Le gestionnaire de projet est une interface qui possède un éditeur de textes où se fait l'implémentation du système. La compilation (passage de JAL à Java) et l'exécution du système se font aussi à l'intérieur de cette interface.

A.4.2.6. MadKit

La plateforme MadKit[11] développée à l'Université de Montpellier II, est basée sur la méthodologie Aalaadin [FER 98], la notion d'agent, de groupe et de rôle. MadKit fournit une Api permettant la construction d'agents en spécifiant une classe d'agent abstraite. Chaque agent peut tenir différents rôles au sein de différents groupes. Les agents sont lancés par le noyau de MadKit. L'outil fournit un éditeur permettant le déploiement et la gestion des SMA (a-box). La gestion faite via cet éditeur offre plusieurs possibilités intéressantes. Il est ainsi possible d'échanger des messages directement avec un agent ou avec un groupe d'agents. Cette plateforme est surtout intéressante pour l'approche organisationnelle qu'elle met en avant lors de l'analyse et de la conception d'un système multi-agents. L'outil offre aussi un utilitaire pour effectuer des simulations.

A.4.2.7. DIMA

DIMA (Développement et Implémentation de Systèmes Multi-Agents) est un environnement de développement de SMA qui a été créé dans le cadre de la thèse de Z. Guessoum[12]. DIMA propose des agents réactifs, cognitifs, hybrides, autonomes, adaptables. La première version de DIMA a été implémentée en utilisant le langage Smalltalk-80 et a été ensuite portée en JAVA. Il fournit des librairies offrant les briques de base pour construire des modèles d'agents divers. Les langages de communication entre agents sont basés sur les langages KQML et ACL de la FIPA. La plateforme DIMA est destinée aux simulations et aux résolutions de problèmes. Elle a été utilisée pour développer plusieurs applications réelles comme la ventilation artificielle [DOJ 97], la simulation des modèles économiques [DUR 00], la simulation d'un marché électrique, etc.

A.4.3. Plateformes pour la mobilité

Pour concevoir et implémenter une plateforme d'agents mobiles, on distingue trois approches. La première consiste à trouver un langage de programmation qui comprend des instructions pour les agents mobiles. La deuxième est d'implémenter le système d'agents mobiles comme des extensions du système d'exploitation. Et la troisième approche construit la plateforme comme une application spécialisée qui tourne au-dessus d'un système d'exploitation [SAM 03]. La plupart des systèmes utilisent cette dernière approche qui se resume souvent en une collection de librairies Java comme Voyager, Aglet, Concordia, Mole, Odyssey. A titre d'exemple, nous présentons dans Tableau 4 quelques plateformes représentatives des systèmes pour agents mobiles.

Produit	Institut de recherche	Langage	Description
Aglets	IBM Japan	Java	Agents Mobiles
Concordia	Mitsubishi Electric	Java	Agents Mobiles
Grasshopper	IKV++	Java	Agents Mobiles
GenA	CRIM	GenA	Plateforme d'Agents Mobiles

Tableau 4. *Plateformes pour la mobilité*

[11] http://www.madkit.org/
[12] http ://www-poleia.lip6.fr/~guessoum/dima.html

A.4.3.1. Grasshoper

Grasshoper[13] est une plateforme qui fournit la gestion de la migration d'agents et la continuité des communications lors de ces migrations. Les services fournis par la plateforme sont la gestion du cycle de vie des agents, la gestion d'annuaires et la migration.

A.4.3.2. Aglets

Aglets est une plateforme developpée par IBM Japon pour définir un cadre de développement d'agent mobile [SAM 03]. Dans cette optique, la plateforme prend en charge les services nécessaires à la migration d'agents et à la gestion des communications. Une aglet[14] est une classe Java étendant une classe fournie par le noyau et redéfinissant les fonctions à activer lors des sérialisations/dé-sérialisations des agents. L'architecture d'un aglet ressemble beaucoup à celle d'un applet Java et le système a son propre protocole pour le transfert des aglets entre hôtes qui est le *Aglet Transfer Protocol*.

A.5. Bibliographie

[BOI 01] BOISSIER O., Cours SMA-DEA-CCSA Multi-Agent systems, MAS Platforms, SMA/SIMMO, EMNS Ecole des Mines de Saint-Etienne, 5 décembre 2001

[BRA 96] BRADSHAW J., « KAoS: An Open Agent Architecture Supporting Reuse, Interoperability, and Extensibility », Research and Technology, Boeing Information and Support Services, Seattle, 1996.

[CAL 99] CALDERONI S., « GEAMAS », Thèse de Doctorat de l'Université de la Réunion, Novembre 1999.

[DOJ 97] DOJAT M., PACHET F., GUESSOUM Z., TOUCHARD D., HARF A., BROCHARD L., « NéoGanesh : a Working System for the Automated Control of Assisted Ventilation in ICUs », Artificial Intelligence in Medicine, vol.11, n°2, septembre-octobre 1997.

[DUR 00] DURAND R., GUESSOUM Z., « Competence systemics and survival, simulation and emperical analysis », dans Competence 2000 Conference, Helsinki, Finlande, p. 10-14, juin 2000.

[FER 89] FERBER J., GUTKNECHT O., « Alaadin : a meta-model for the analysis and design of organizations in multi-agent systems », ICMAS'98, p. 128-135, Paris, 1998.

[GAR 02] GARNEAU T., « Programmation orientée-agent : évaluation comparative d'outils et environnements », Systèmes multi-agents et systèmes complexes, JFIADSMA, Lille, France, 2002.

[GUE 01] GUESSOUM Z., ''Principes et architecture des systèmes multi-agents'', sous la direction de Jean-Pierre Briot et Yves Demazeau, Chapitre 5, 2001.

[JIM 96] WHITE J., General Magic, Mobile Agents White Paper, General Magic, copyright 1996.

[MAR 02] MARIETTO M.B., DAVID N., SICHMAN J.S., COELHO H., Requirements Analysis of Agent-Based Simulation Platforms. Technical Report, 2002.

[MAT 01] MATHIEU P. and ROUTIER J.C., « Une contribution du multi-agent aux applications de travail coopératif », TSI Hermès Science Publication, Numéro Spécial : Télé-applications, 2001.

[NUN 02] NUNO D., MARIETTO M.B., SICHMAN J.S., HELDER Coelho. « *Requirements Analysis of Multi-Agent-Bases Simulation Platforms: State of the Art and New Prospects*», 2002.

[PIE 03] PIERRE S., « Piranhas 2, Concepts et réalisation », *24 janvier 2003*

[RIC 00] RICORDEL P. M. and DEMAZEAU Y., "From Analysis to Deployment : a MultiAgent Platform Survey", 1st International Workshop on Enginnering Societis in the Agents World (ESAW), ECAI'2000, A. Ominici, R. Tolksdorf and F. Zambonelli (eds.), Berlin, Allemagne, Aout 2000.

[SAB 01] SABAS A., « Systèmes multi-agents : une analyse comparative des méthodologies de développement, vers la convergence des méthodologies de développement et la standardisation des plateformes SMA », Université du Québec à Trois-Rivières, Octobre 2001.

[SAM 03] SAMUEL P., « Réseaux et systèmes informatiques mobiles, Fondements, architectures et applications », Presse internationale Polytechnique, 2003.

[13] Grasshoper-2 Agent Development Platform : http://www.grasshopper.de
[14] http://www.aglets.org ou http://www.trl.ibm.com/aglets

[SEC 03] SECQ Y., « RIO: Rôles, Interactions et Organisations, une méthodologie pour les systèmes multi-agents ouverts », Université des Sciences et Technologies de Lille, rapport de thèse soutenue le 2 Décembre 2003.

[SIM 01] Simulation of Cognitive Agents, « Requirements Analysis of Multi-Agent Based Simulation Platforms, Simulation of Cognitive Agents », rapport technique, 2001.