Institut de la Francophonie pour l'Informatique

Laboratoire Informatique de l'Université de Pau et des Pays de l'Adour

MEMOIRE DE STAGE DE FIN D'ETUDES
MASTER EN INFORMATIQUE

Sujet :

Vers un développement distribué efficace des processus d'adaptation de documents multimédias

Stagiaire : Pham Hai Quang

Responsables de stage : Sébastien Laborie
Philippe Roose

Ce stage s'est déroulé au sein du LIUPPA sur le site d'Anglet (Parc Montaury) à l'IUT de Bayonne de l'UPPA

Anglet, Mai – Septembre, 2011

Remerciements

Je tiens particulièrement à remercier mes encadrants Philippe Roose et Sébastien Laborie, pour leur aide et conseils précieux durant ces 5 mois de stage.

J'adresse mes sincères remerciements à tous les professeurs de l'Institut de la Francophonie pour l'Informatique (IFI) pour m'avoir enseigné et dispensé les cours intéressants pendant mes études au niveau master. Je profite également de cette occasion pour dire remercier à tous les personnels de l'IFI qui m'ont apporté de l'aide.

En fin, je voudrais remercier ma famille, mes parents et mes amis qui sont toujours près de moi et m'ont apporté du courage dans les moments difficiles.

Résumé

De nos jours, avec le développement de l'Internet et des technologies, beaucoup de dispositifs peuvent actuellement accéder à multiples informations et utiliser des services crées pour Smartphone, tablette... Donc, on doit faire face aux problématiques de compatibilité : compatibilité du format des données avec les formats supportés par le dispositif utilisé, compatibilité des données transmises vis-à-vis de la capacité du réseau et du dispositif utilisé, ergonomie de l'affichage sur le dispositif utilisé.

L'adaptation de document multimédia est considérée comme une solution efficace et intéressante à cette problématique. Elle consiste en la transformation du document multimédia afin qu'il soit compatible avec les contraintes du contexte d'exploitation.

Dans mon stage, je propose un architecture d'adaptation de document multimédia. Cette architecture constitue une solution ouverte et générique pour les problèmes d'adaptation de document multimédia qui est exécutée sur le serveur multimédia. L'idée de base de cette solution est d'exécuter un algorithme de recherche pour trouver un service ou une chaîne de services qui adapter des contraintes d'utilisateur. Les services sont distribués sur des serveurs multimédias, donc je propose aussi un protocole pour transférer des données entre les serveurs.

Mots-clés : Systèmes pervasifs, Adaptation au contexte, Qualité de Service, Documents et contenus multimédias, Mobilité (téléphone portable, PDA…)

Abstract

Today, with the development of the Internet and technology, many devices can access information and use multiple services created for Smartphone, tablet ... So we must face the problem of compatibility: the data format compatibility with the formats supported by the device used, compatibility of data transmitted vis-à-vis the capacity of the network and the device used and ergonomics of the display on the screen of device used.

The adaptation of multimedia document is considered an effective and interesting to this problem. It is the transformation of the multimedia document to make it compatible with the constraints of the context exploitation.

In my training period, I propose an architecture for multimedia document adaptation. This architecture is an open solution for the generic problems of adaptation of multimedia document that is executed on the media server. The basic idea of this solution is to run an algorithm to find a service or a chain of services that solve user constraints. Services are distributed on media servers, so I also propose a protocol for transferring data between servers.

Keywords: pervasive systems, context adaptation, quality of service, document and multimedia content, mobility (mobile phone, PDA,...)

Table des matières

Table des figures

Liste des tables

Liste des cadres

Chapitre 1 : Introduction

1.1. Problématique

De nos jours, Internet est très présent dans nos quotidien. En effet, on peut lire des informations partout sur le monde avec un ordinateur qui se connecte à l'Internet. Avec le développement de l'Internet et des technologies, beaucoup de dispositifs peuvent actuellement accéder à multiples informations et utiliser des services crées pour Smartphone, tablette...Ces dispositifs entraînent des usages différents des services de l'Internet. Par exemple, un étudiant regarde une vidéo sur l'Internet quand il rentre chez lui. L'étudiant peut utiliser son Smartphone pour visualiser cette vidéo quand il est sur la rue. Puis, quand il prend le train, il peut utiliser sa tablette pour disposer d'un plus grand affichage. Il se connecte sur le réseau 3G lorsqu'il est dans la rue et dans le train. Quand il arrive à son appartement, il utilise son ordinateur portable qui se connecte au réseau wifi. L'ordinateur portable donne la meilleur qualité parce que la vitesse de l'Internet est plus importante, plus stable, et ce dispositif dispose d'un haute résolution d'écran, le matériel est très puissant...

L'exemple ci-dessus illustre la mobilité de l'utilisateur, les dispositifs utilisés et les types variés de réseaux. Dans ce scénario, nous mettons en évidence des problèmes : la compatibilité du format de données avec les formats supportés pour les dispositifs utilisés, la compatibilité de la taille des données avec la capacité des réseaux et le problème de la résolution des dispositifs.

Il n'est plus à démontrer que l'utilisation des appareils mobiles est devenue incontournable et avec eux l'utilisation de services communément appelés « apps », ce qui ne va pas sans poser des problèmes lors de la transmission de contenus (i.e., image, texte, audio, vidéo) et plus généralement de documents multimédias comme les MMS.

Dans le même temps, on assiste aujourd'hui à un foisonnement des appareils mobiles de type iPhone, Black Berry, Smartphone ou « générique » tournant par exemple sous Android ceci impliquant une grande diversité et hétérogénéité des dispositifs en terme de caractéristiques techniques (e.g., CPU, RAM, batterie) et physiques (e.g., taille d'écran, GPS, son, capacités d'interaction).

Par conséquent, la diffusion de documents multimédias sur ce type d'appareil n'est pas une chose aisée. En effet, pour qu'un document soit exécuté sur de multiples dispositifs celui-ci doit satisfaire de multiples contraintes spécifiées dans des profils provenant de

caractéristiques physiques (comme la taille d'écran par exemple), des souhaits ainsi que des préférences des utilisateurs, de leur environnement (e.g., en voiture, en réunion, à la maison), etc. Il est encore plus difficile de prendre en compte les caractéristiques techniques (batterie restante, RAM disponible, etc.) qui vont guider la politique des adaptations en fonction des ressources disponibles à l'instant t des appareils, conjointement aux adaptations nécessaires.

Durant mon stage, j'ai spécifié une architecture d'adaptation ainsi qu'un algorithme de recherche de services qui sont nécessaire à l'adaptation. L'idée de base est la suivante:

- L'utilisateur demande une ressource multimédia sur un serveur multimédia, il envoie des contraintes de son contexte(la capacité du dispositif, la capacité du réseau...) au serveur multimédia.
- Le système d'adaptation analyse les contraintes de l'utilisateur. Puis, il cherche des services d'adaptation qui permettent de transformer la ressource multimédia pour satisfaire ces contraintes. Enfin, il exécute les services pour adapter la ressource et revoie la ressource adaptée à l'utilisateur.

1.2. Organisation du mémoire

Ce mémoire vise à rapporter ce que j'ai fait pendant mon stage au sein de l'équipe LIUPPA sous la direction des professeurs Philippe Roose et Sébastien Laborie.Le rapport est structuré en 5 chapitre :

- Le premier chapitre est consacré à l'introduction du sujet de stage.
- Dans la deuxième chapitre, je présente les approches existants pour adapter le document multimédia.
- Le troisième chapitre, je propose un architecture d'adaptation, le schéma d'adaptation et l'algorithme de recherche des services d'adaptation.
- Le quatrième chapitre, je présent l'implémentation et l'évaluation de mon architecture.
- Le cinquième chapitre conclut mon rapport et les perspectives

Chapitre 2: État de l'art

2.1. Définitions

Pour clarifier les notions fondamentales et le vocabulaire qui intervient dans le domaine de l'adaptation, nous définissons plusieurs termes qui seront utilisés dans ce rapport.

Objet multimédia: Un objet multimédia est une entité qui fait référence à une ressource pouvant être, par exemple, de l'audio, du texte, une image ou bien une vidéo.

Document multimédia: Un document multimédia est constitué d'un ensemble d'objets multimédia. Ceux-ci sont mis en page grâce à des techniques d'assemblage propres à l'auteur et forment la composition du document multimédia.

Présentation multimédia: Une présentation multimédia correspond à une exécution du document sur une plate-forme et à un instant donné.

2.2. Approches existantes

Nous allons étudier dans la suite les différents approches existants permettant d'adapter le document multimédia. Trois approches élémentaires ont été proposées pour la localisation des processus d'adaptation sur le parcours de transmission des données, entre la source des données (serveur multimédia) et la destination des données (client final) : (i) au niveau de la source, (ii) au niveau du destinataire ou client et (iii) au niveau d'un intermédiaire (proxy)[6,7,8,9,10].

Dans la première approche, au niveau de la source ou approche centrée serveur, le serveur se charge de découvrir les capacités du client et la bande passante disponible. Il décide alors de la meilleure stratégie d'adaptation. Un auteur peut pré-visualiser et valider le résultat de l'adaptation selon différentes conditions. Cependant, ces approches présentent deux inconvénients majeurs : d'une part, elles imposent aux fournisseurs de document multimédia d'intégrer des mécanismes d'adaptation ; d'autre part, les processus de transformation induisent une charge de calcul et une consommation de ressources importantes sur le serveur que celui-ci n'a souvent pas la capacité de fournir.

Dans la seconde approche, l'adaptation est réalisée au niveau du destinataire. On appelle aussi l'approche centrée client. Le processus d'adaptation est effectué par le terminal client, en fonction des capacités de celui-ci. Le serveur demeure donc inchangé et transmet ses documents multimédias dans leur format d'origine. Le terminal du client utilisateur, à la réception du document multimédia, transforme celui-ci de manière à pouvoir l'afficher correctement. L'adaptation centrée client requiert des modifications minimales au niveau du

terminal de connexion et peut être extrêmement efficace dans la personnalisation du document multimédia et l'adaptation aux caractéristiques du terminal. Toutefois, la complexité souvent élevée des mécanismes d'adaptation interdit l'utilisation générique de cette solution car la limitation de puissance de calcul, de mémoire, d'énergie et de stockage de dispositif utilisé. Ainsi, les approches centrées client ne sont en pratique pertinentes que vis-à-vis de problématiques simples d'affichage lorsque les contraintes réseaux sont traitées par ailleurs.

La troisième approche tente de constituer un compromis entre les deux solutions précédentes, l'adaptation est ainsi exécutée sur un nœud intermédiaire habituellement qualifié de proxy. Dans cette approche, le proxy intercepte la réponse du serveur, décide et effectue l'adaptation en fonction des préférences de l'utilisateur, des capacités des terminaux et de l'état du réseau, puis envoie le document adapté au client. Cette solution présente plusieurs avantages. Premièrement, la charge de calcul du serveur multimédia est déplacée vers le proxy. Deuxièmement, le proxy peut être positionné au point le plus critique du chemin de données. On considère habituellement que la bande passante entre un proxy et un serveur est plus large que celle qui est disponible entre un client (souvent mobile) et un proxy. Le placement sur le dernier maillon de la chaîne permet au proxy de disposer d'une vue globale sur les ressources de l'environnement, telles que la latence du réseau, la bande passante, la taille du document à transporter avant la livraison finale, les préférences de l'utilisateur, toute modification survenant dans le temps... Le serveur proxy joue aussi un autre rôle majeur de mise en cache des résultats de l'adaptation de documents multimédias afin d'éviter des allers-retours vers le serveur multimédia et la répétition d'opérations de transcodage coûteuses lorsque les ressources peuvent être accessibles depuis le cache. Les approches orientées proxy présentent aussi des inconvénients. En premier lieu, elles passent mal à l'échelle du fait de coûts de calcul considérables induits par les opérations d'adaptation imposant des unités de calcul puissantes et beaucoup de mémoire. En second lieu, le fait de passer par un intermédiaire pour la transmission d'un document soulève des problèmes de sécurité. Le tiers manipulant le proxy doit être digne de confiance auprès de l'utilisateur et de la source. Enfin, la liste des outils d'adaptation de document multimédia n'est pas exhaustive et est amenée à évoluer ; l'intégration de ces nouveaux outils risque de ne pas être possible si le proxy n'est pas fonctionnellement extensible.

En résumé, on a mentionné les approches différentes d'adaptation de document multimédia. Ces solutions ont développé des stratégies, des architectures d'adaptation. Cependant aucune satisfait les problématiques d'extensibilité, de flexibilité. En outre, les types d'adaptation de

document multimédia sont principalement focalisés sur la transformation d'image, et ne fournissent pas de solution générique d'adaptation. Durant mon stage, je propose une architecture extensible et flexible. Dans cette architecture, la stratégie d'adaptation est exécutée par serveur multimédia ou bien le serveur d'annuaire de services. Le serveur d'annuaire de services décide les services à adapter et les serveurs les exécutent. Les services sont distribués dans les serveur multimédias, donc les serveurs doivent se communiquer par un protocole. Cette architecture peut aussi élargir le nombre de services d'adaptation pour adapter le document multimédia dans beaucoup de contextes. De plus, l'utilisateur peut demander le document multimédia correspondant à n'importe quel contexte et à n'importe quel dispositif cible.

Chapitre 3 : Proposition

3.1. L'architecture générale de mon processus d'adaptation

L'architecture générale est montrée dans la figure 3.1 :

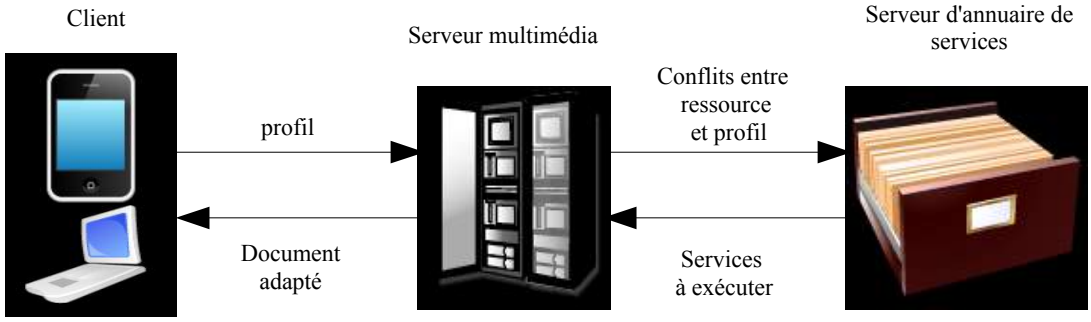

Figure 3.1: Architecture générale

Le modèle général consiste en 3 composants :

- Client : c'est le dispositif que l'utilisateur utilise. Ce sont des ordinateurs fixes, des ordinateurs portables, tablettes, téléphones...
- Serveur multimédia : il enregistre des documents multimédias. Il stocke et exécute aussi des services d'adaptation.
- Serveur d'annuaire de services : il enregistre des descriptions de services adaptés des serveurs multimédias.

Dans le modèle général, on utilise l'Internet pour accéder aux documents multimédias qui sont enregistrés sur un serveur. Quand le client demande un document multimédia, il envoie un profil en fichier XML qui représente des caractéristiques du dispositif utilisé . Le serveur compare les caractéristiques reçues et les caractéristiques du document multimédia demandé pour obtenir des descriptions de conflit et ainsi déduire les adaptations à appliquer. Le serveur enregistre des descriptions de services d'adaptation pour avoir connaissances des services de transformation disponibles. Basé sur les descriptions de problèmes et les descriptions de services, le serveur exécute un algorithme de recherche de services d'adaptation pour fournir une chaîne de services qui permet d'adapter le document multimédia. Ensuite, il exécute ces services. Enfin, il renvoie le document adapté au client. Quand le serveur recherche des services, il peut y avoir les situations décrites dans le diagramme dans la figure 3.2:

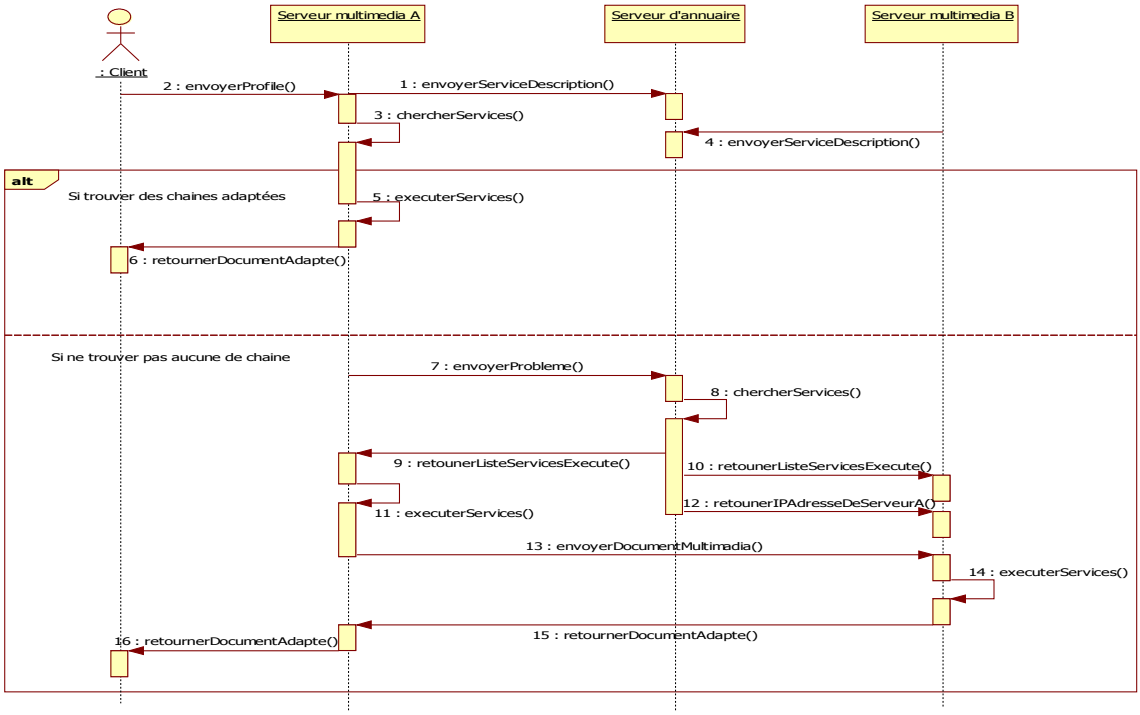

Figure 3.2 : Diagramme de la recherche des services

Dans ce qui suit, j'explique les situations du diagramme en détail :

1. Dans un premier temps, le serveur essaye de trouver des services locaux (étape 3). S'il trouve plusieurs résultats, c'est-a-dire il existe des services ou des chaînes de services locaux qui satisfont les problèmes d'adaptation. Dans ce cas, on utilise un paramètre appelé le poids. A chaque service est attribué un poids. La valeur du poids de chaque service dépend des caractéristiques de ce service. Par exemple, la valeur du poids est calculée via la qualité de service comme le temps d'exécution, l'usage de ressources, qualité du résultat... Le serveur sélectionne la chaîne qui va être exécutée selon le poids total de la chaîne. La chaîne ayant le poids total minimal est choisie.
2. Si le serveur ne trouve pas de service ou de chaîne de services locaux pour satisfaire les problèmes d'adaptation, dans ce cas, on utilise un serveur d'annuaire de services (étape 8). Il contient toutes les descriptions des services d'adaptation des serveurs multimédias. Pour avoir toutes ces descriptions de services des serveurs multimédias, quand un serveur multimédia est actif, il envoie un fichier en format XML qui décrit les descriptions de services au serveur d'annuaire. Ainsi quand un serveur multimédia ne trouve aucun services ou bien une chaîne de services satisfaisant les problèmes, il envoie les descriptions de problèmes d'adaptation au serveur d'annuaire de services. Le

serveur d'annuaire de services exécute alors un algorithme de recherche qui est le même que celui utilisé dans le serveur multimédia pour chercher des chaînes de services d'adaptation. Basé sur le poids total, le serveur d'annuaire de services choisit une chaîne optimisée. La chaîne de services peut être composée de services qui appartiennent à des serveurs multimédias différents. Ensuite, le serveur d'annuaire de services envoie les adresses IP des serveurs qui exécutent les services dans la chaîne au serveur multimédia qui le consulte. Le serveur d'annuaire de services envoie aussi les identités des services dans la chaîne aux serveurs où ils sont exécutés. Enfin, les serveurs exécutant les services dans la chaîne reçoivent la ressource multimédia pour exécuter les services et après, ils envoient les résultats au serveur ayant les ressources multimédia.

Nous avons fait ce choix de trouver tout d'abord des services locaux pour des raisons d'efficacité. En effet, chaque serveur multimédia dispose d'un nombre moins important de services que sur le serveur d'annuaire qui connaît tous les services d'adaptation de tous les serveurs multimédias.

3.2. Le modèle fonctionnel du système d'adaptation

La fonction du système d'adaptation est de fournir à l'utilisateur un document multimédia satisfaisant son contexte d'usage comme les caractéristiques du dispositif utilisé, les caractéristiques de réseaux…Le système d'adaptation est basé sur les problèmes d'adaptation et l'ensemble des services d'adaptation disponibles pour trouver un service ou une chaîne de services qui résolvent les problèmes d'adaptation. Ce système exécute le service ou bien la chaîne de services avec l'entrée qui est le document multimédia pour produire le document multimédia adapté. Le modèle du système d'adaptation est illustré dans la figure 3.3 :

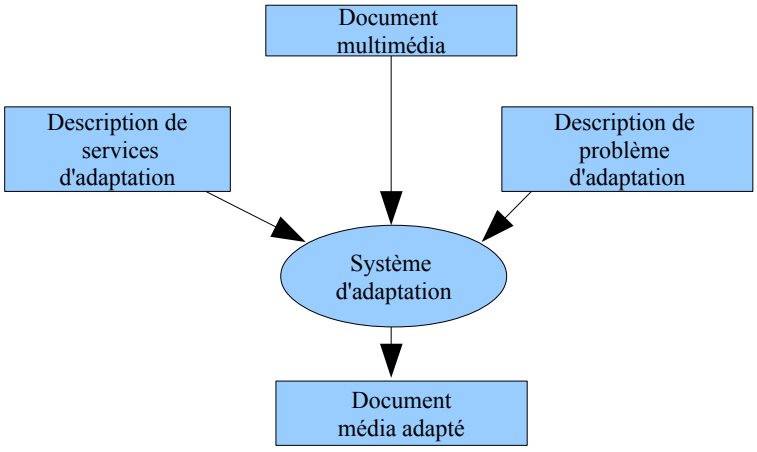

Figure 3.3 : Modèle fonctionnel

- Document multimédia : c'est la ressource à adapter.
- Description de services d'adaptation : il contient les descriptions de services d'adaptation.
- Description de problèmes d'adaptation : il contient les conflits qu'il faut adapter.

3.3. Système d'adaptation

3.3.1. Le module d'adaptation

Le module d'adaptation se compose de trois composants : profil, mécanisme d'adaptation et d'exécution(voir la figure 3.4). Dans ces composants, le composant mécanisme d'adaptation est le plus important car c'est lui qui adapte en transformant le document multimédia. Ce composant cherche la meilleure chaîne de services d'adaptation pour satisfaire les caractéristiques du profil. Le mécanisme d'adaptation exécute un algorithme de recherche des chaînes de services d'adaptation en se basant sur les paramètres suivants : les caractéristiques du profil, les caractéristiques des ressources et les services disponibles. S'il n'y a pas de chaîne de service, on consulte le serveur d'annuaire de services. Le serveur d'annuaire de services exécute le même algorithme qu'on implémente dans le composant du mécanisme d'adaptation . Après avoir trouvé la meilleure chaîne, le composant d'exécution exécute tous les services de la chaîne. Les services dans la chaîne sont ordonnés. La sortie d'un service est l'entrée du prochain service. Quand tous les services dans la chaîne sont exécutés, on obtient le document adapté. Les composants du module d'adaptation sont montrés dans la figure 3.4. Dans ce diagramme, les boites continues représentent les composants et les boites en pointillé représentent les sorties.

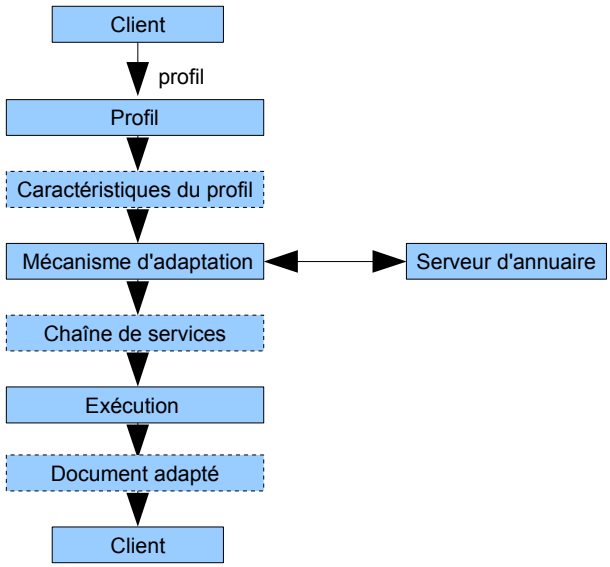

Figure 3.4 : Module d'adaptation

3.3.2. Le schéma d'adaptation

Selon le module d'adaptation qui est représenté dans la section 3.3.1, je construis le processus d'adaptation. Le composant « profil » dans le module est implémenté par les fonctions : recevoir le fichier du profil et extraire les caractéristiques de ce profil. L'extraction des caractéristiques du profil est exécutée après avoir reçu le fichier du profil et l'avoir stocké dans le système. Une fois les caractéristiques du profil déterminées, on extrait les caractéristiques du document multimédia. Ensuite, on fait la recherche des problèmes(ou conflits) en comparant les caractéristiques du profil avec celles de la ressource média. L'extraction des caractéristiques de ressources et la comparaison appartiennent au composant « adaptation mécanisme ». Dans ce composant, on implémente en plus deux taches : l'extraction des descriptions de services et la recherche de services qui font l'adaptation. On exécute exclusivement l'extraction des descriptions de services s'il existe des conflits entre le profil et le média. S'il n'existe aucun conflit, on renvoie immédiatement le document multimédia au client. Dans ce composant, la recherche des services est la plus importante. Je présenterai dans les parties suivantes un algorithme complexe pour la recherche. Le composant « exécution » sélectionne dans l'ensemble des chaînes d'adaptation la meilleure chaîne, c'est-à-dire celle qui dispose d'un poids minimal. Après avoir sélectionné cette chaîne, on exécute successivement les services de cette chaîne. Le schéma est décrit dans la figure 3.5 dans lequel les boites continues représentent les processus, les boites entrecoupées représentent les données.

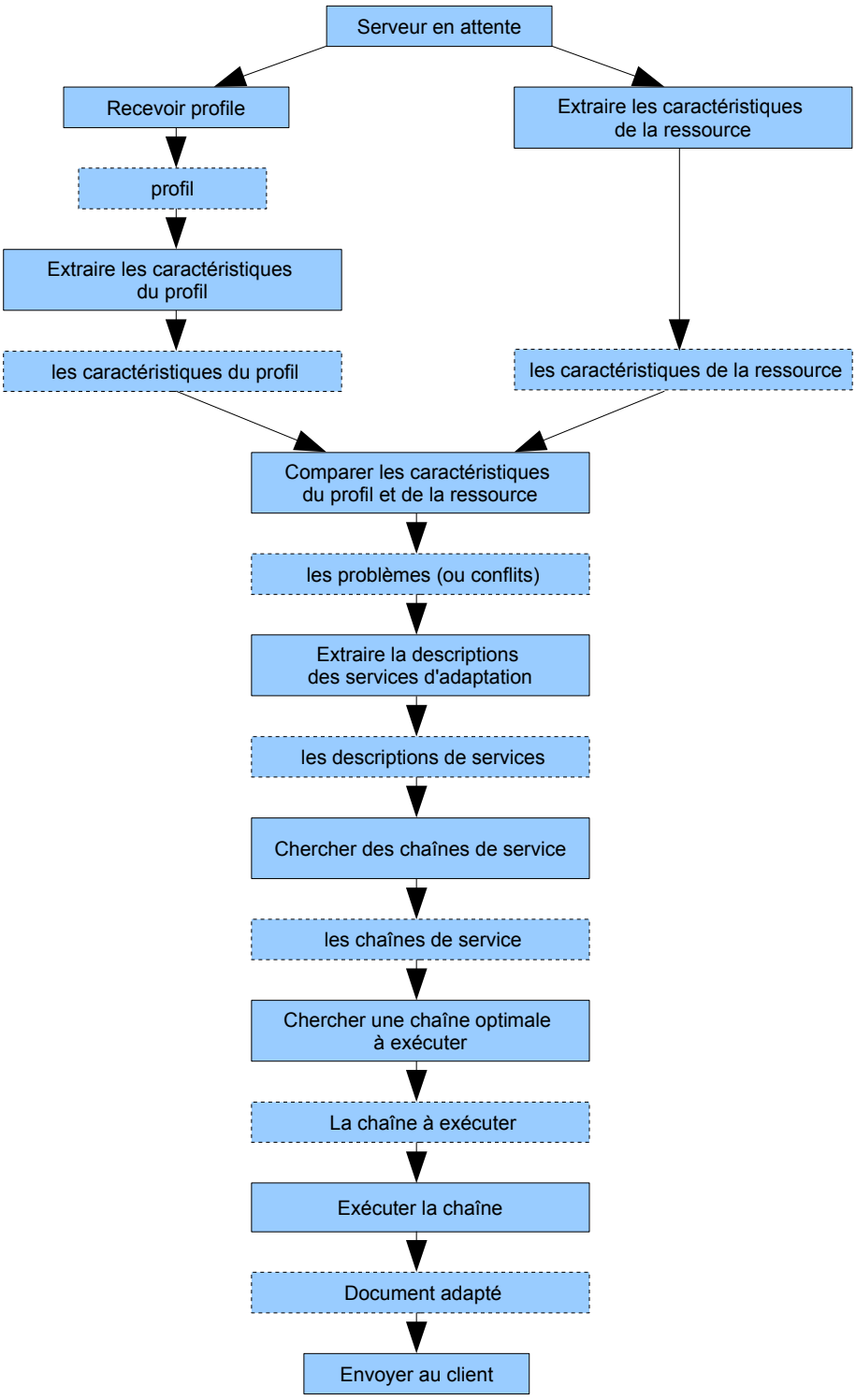

Figure 3.5 : Le schéma d'adaptation

Dans les sou-section suivantes, j'illustre le schéma en détail.

3.3.3. Extraction des caractéristiques du profil

Le client décrit ses capacités logicielles et matérielles au sein d'un profil: la résolution, le

format, la taille,... Ce profil est décrit dans un ficher XML. Ensuite, il l'envoie à un serveur. Voici un exemple de profil en format XML comme dans la cadre 3.1:

```
<?xml version="1.0" encoding="UTF-8" ?>
<profil>
    <media  type = "image"
            format = ".png"
            resolution = "1200x800" />
</profil>
```

Cadre 3.1 : Contenu du profil

Comme il est décrit dans l'exemple ci-dessus, le dispositif a besoin de recevoir une image en format PNG avec la résolution de 1200x800 pixels. D'autres caractéristiques du profil peuvent être décrites comme la taille de l'écran, la bande passante... Celles-ci ne sont pas représentés dans ce fichier, c'est-a-dire le client accepte n'importe quelle valeur de ces caractéristiques. On utilise les caractéristiques pour faire une description de profil, comme illustré dans la table ci-dessous:

Paramètre	Valeur	Description
mediaType	"image"	Le type multimédia
format	".PNG"	Le format que le client accepte
resolution	"1200x800"	La résolution que le client accepte

Table 3.1 : Caractéristique du profil

Le dispositif peut accepter plusieurs valeurs de chaque caractéristique. Par exemple : l'utilisateur peut accepter les images en format PNG, GIF. Ainsi, la valeur de chaque caractéristique peut-être un ensemble de valeurs. Dans l'exemple, on a une description de profil qui autorise plusieurs format d'affichage d'image comme dans la table 3.2:

Paramètre	Valeur	Description
mediaType	"image"	Le type multimédia
format	".PNG,.GIF"	Le format que le client accepte
resolution	"1200x800"	La résolution que le client accepte

Table 3.2 : Caractéristique du profil

3.3.4. Extraction des caractéristiques du document multimédia

Pour trouver des problèmes d'adaptation, il faut avoir les caractéristiques du document multimédia à transmettre. Dans la description de la ressource, les noms des paramètres sont les mêmes que ceux dans la description du profil. Par exemple, le média est une image avec un format JPEG et une résolution de 1900x1020:

Paramètre	Valeur	Description
mediaType	"image"	Le type de la ressource
format	".JPG"	Le format de la ressource
resolution	"1900x1020"	La résolution de la ressource
size	"4000"	La taille de la ressource en octets
….		

Table 3.3 : Caractéristique de la ressource

3.3.5. Recherche des problèmes

Après les deux étapes d'extraction des caractéristiques de profil et de ressource, on a deux descriptions : une du profil et une autre sur la ressource. On fait la comparaison entre ces deux descriptions pour obtenir les problèmes en comparant les paramètres. Si deux paramètres ont même nom et si les valeurs des ressources n'existent pas dans la description du profil, on détecte un problème. Dans la description du problème, on représente des paramètres comme : contentFormat, contentValeur, profilValeur, mediaType, actionType. Ces paramètres fournissent les conditions pour trouver les services d'adaptation. Selon les descriptions des deux exemples ci-dessus, on trouve deux problèmes : problème de format et problème de résolution. On montre les descriptions de ces deux problèmes dans les deux tables ci-dessous:

Paramètre	Valeur	Description
contentFormat	".JPG"	Le format de ressource
contentValeur	".JPG"	La valeur de format de ressource
profilValeur	".PNG"	Le format que l'utilisateur demande
mediaType	"image"	Le type de la ressource
actionType	"transcodage"	Le type d'action

Table 3.4 : Description du problème

Paramètre	Valeur	Description
contentFormat	".JPG"	Le format de ressource
contentValeur	"1900x1020"	La valeur de format de ressource
profilValeur	"1200x800"	Le format que l'utilisateur demande
mediaType	"image"	Le type de la ressource
actionType	"transformation"	Le type d'action

Table 3.5 : Description du problème

Dans ces tables, on voit le paramètre « actionType » avec les valeurs : « transcodage » et «transformation ». On divise les types d'adaptation en 3 cas :

- Le changement de format en même type de document multimédia , par exemple : quand on change le format d'une image du JPG au PNG. L'image d'entrée est de type « image », et le type de l'image sortie est identique. Dans ce cas, la valeur du paramètre « actionType » est « transcodage ».
- Le changement de format où le format de sortie et celui de l'entrée sont différents. Par exemple, dans le service TextToSpeech, on change un texte en son. Dans ce cas, la valeur du paramètre « actionType » est « transmodage ».
- Il n'y a pas le changement de format. Par exemple : quand on change la taille d'image de format JPG, la sortie est une image de format JPG avec la nouvelle taille. Dans ce cas, la valeur du paramètre « actionType » est « transformation ».

3.3.6. Extraction de la description des services d'adaptation

Chaque serveur multimédia enregistre une représentation en format XML des services que ce serveur héberge. Chaque service est présenté comme un élément. Chaque élément contient des attributs représentés dans la table ci-dessous :

Attribut	Description
id	L'identité de service
name	Le nom de service
input	L'entrée de service
output	La sortie de service
className	La classe qui exécute le service
mediaType	Le type de document multimédia auquel ce service applique
actionType	Le type d'action de service
weight	Le poids de service

Table 3.6 : Description du service

Les attributs dans la table représentent les caractéristiques d'un service. Les attributs comme : input, output, mediaType et actionType, nous aident à chercher un service ou une chaîne de services pour résoudre un problème d'adaptation. Basé sur les résultats de la recherche, on utilise l'attribut « weight » pour sélectionner la meilleure chaîne si plusieurs solution d'adaptation sont possibles. Quand la meilleure chaîne est sélectionnée, le serveur

d'annuaire de services d'adaptation envoie les identités des services aux serveurs multimédias qui les exécutent. On utilise l'attribut « name » pour afficher le nom des services. Le cadre 3.2 présente un exemple de description de services en format XML :

```
<?xml version="1.0" encoding="UTF-8" ?>
<services>
    <service  id = "1"
              name = "ConvertImageJPG2PNG"
              input = ".jpg"
              output = ".png"
              className = "ConvertImageJPG2PNG.java"
              mediaType = "image"
              actionType = "transcodage"
              weight = "2" />
    <service  id = "2"
              name ="ConvertImagePNG2JPG"
              input = ".png"
              output = ".jpg"
              className = "ConvertImagePNG2JPG.java"
              mediaType = "image"
              actionType = "transcodage"
              weight = "2" />
    <service  id = "3"
              name ="Resize"
              input = ".png"
              output = ".png"
              className = "Resize.java"
              mediaType = "image"
              actionType = "transformation"
              weight = "1" />
   <service  id = "4"
              name ="TextToSpeech"
              input = ".txt"
              output = ". 3gp"
              className = "T2S.java"
              mediaType = "text"
              actionType = "transmodage"
              weight = "3" />
</services>
```

Cadre 3.2 : Description de services en format XML

Après l'extraction de la description des services en format XML, on obtient les services avec leurs caractéristiques ainsi que la valeur de chaque caractéristique. Ci-joint, un exemple de description d'un service :

Paramètre	Valeur	Description
ID	"1"	L'identité de service. Dans un serveur multimédia, il n'y a pas deux mêmes IDs
name	"ConvertImageJPG2PNG"	Le nom du service
input	".JPG"	L'entrée du service est l'image en format PNG
output	".PNG"	La sortie de service est un fichier en format PNG
className	"ConvertImageJPG2PNG.java"	La classe qui exécute le service
mediaType	"image"	Le type de document multimédia auquel ce services applique
actionType	"transcodage"	Le type d'action du service
weight	2	Le poids du service

Table 3.7 : Exemple de description d'un service

3.3.7. Algorithme de recherche des services d'adaptation

Dans un scénario classique, l'utilisateur utilisant un dispositif avec des caractéristiques spécifiques demande un document multimédia au serveur multimédia via le réseau Wifi. Le serveur multimédia transforme le document multimédia selon les caractéristiques du dispositif. Dans le cas simple, le document multimédia sur le serveur multimédia s'accorde avec les caractéristiques du dispositif. Le serveur multimédia l'envoie donc directement à l'utilisateur(aucune adaptation n'est nécessaire). Dans le cas plus complexe, le document multimédia sur le serveur multimédia ne s'accorde pas avec les caractéristiques du dispositif. Dans ce cas, le serveur multimédia applique un ou une série de services pour transformer le document multimédia avant de l'envoyer à l'utilisateur. Dans cette partie, je vais expliquer mon algorithme de recherche de services d'adaptation.

Le rôle de l'algorithme est de trouver les chaînes de services qui transmettent le document source en document adapté en satisfaisant le profil du dispositif cible. Cette transformation est exécutée successivement au travers des services dans chaque chaîne, c'est-à-dire la sortie d'un service dans la chaîne est l'entrée du service suivant. La relation est appelée la relation entrée-sortie entre des services. Sur la base de cette relation, on peut rechercher une combinaison de service grâce à un algorithme de backtrack. La recherche des services devient alors un parcours des services sur un arbre. Maintenant, on a deux algorithmes de parcours : l'algorithme de parcours en profondeur (DFS) et l'algorithme de parcours en largeur (BFS). Si on veut toutes les solutions, alors la complexité de ces deux algorithmes sont identiques. Je développe mon algorithme de recherche me basant sur l'algorithme DFS.

Dans le processus d'adaptation, après avoir comparé les caractéristiques du profil et ceux de la ressource, on peut constater des problèmes d'adaptation. Dans mon algorithme de recherche, pour chaque problème, on cherche un ensemble de services potentiels le résolvant. Je donne un exemple : on a un problème de format (il faut changer le format JPG de ressource en format adapté PNG). On cherche les services dont la sortie est l'image en format PNG. À supposer, qu'on dispose des services qui satisfont ce changement de format : S1 (change une vidéo en format MOV en l'image PNG), S2 (change une image JPG en image PNG), S3(change une image GIF en image PNG). Ces services sont donnés dans la table ci-dessous :

Nom de service	Description
S1	Convertir une vidéo MOV en image PNG
S2	Convertir l'image de JPG en PNG
S3	Convertir l'image de GIF en PNG
S4	Changer la résolution de l'image PNG

Table 3.8 : Liste de services

En regardant le tableau, on a les services potentiels, c'est-à-dire ceux qui produisent l'image PNG : S1, S2, S3. Ensuite, on parcourt chaque service dans cet ensemble. Si l'entrée de ce service n'est pas la ressource, c'est-a-dire que le service ne permet pas de changer la ressource en document multimédia adapté, il faut chercher les chaînes de services d'adaptation. Dans cet exemple, S2 satisfait le changement de format, donc il est un résultat. Pour trouver tous les résultats, on vérifie S1 et S3. L'entrée de S1 et de S3 sont une vidéo MOV et une image GIF respectivement. Ces entrées ne sont pas une image JPG, c'est-a-dire si on utilise seulement S1 ou S3, on ne résout pas le changement de format. Donc il est nécessaire de déterminer des chaînes. On cherche des chaînes par l'algorithme de recherche. La recherche part de ce service, et finit quand on trouve un service dont l'entrée est la ressource . Cette recherche est rappelée la recherche par avant. Le processus de recherche par avant se base sur DFS. La recherche par avant est appliquée si on détermine les services qui produisent le document adapté comme S1, S3. Le diagramme dans la figure 3.6 montre le processus de recherche des services d'adaptation dans ce cas :

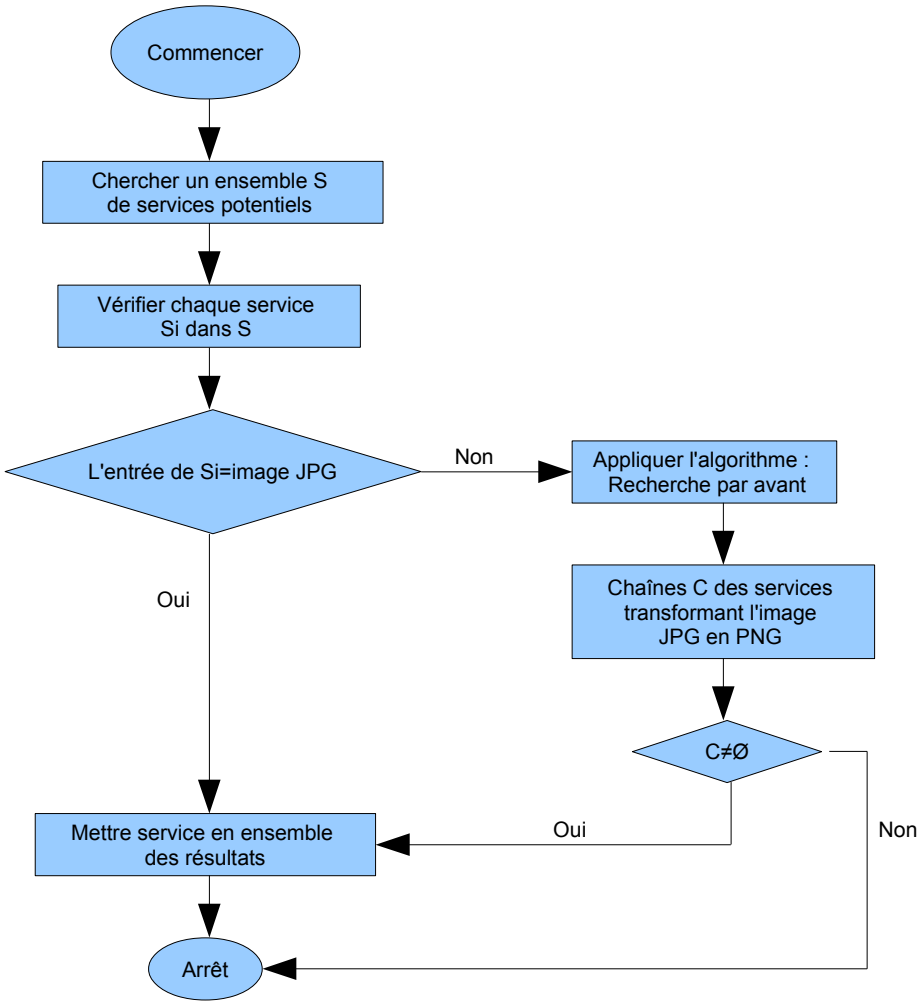

Figure 3.6 : Exemple d'un processus de recherche des services d'adaptation

Dans les autres cas, on utilise la recherche par avant et une autre recherche, c'est la recherche par après. Par exemple : on a un problème de changement de résolution d'une image JPG, mais on a seulement un service S4 qui change la résolution d'une image PNG. L'entrée et la sortie de ce service sont une image PNG. On n'applique donc pas seulement S4 pour adapter le problème. Donc il faut changer l'image JPG en image PNG. Ensuite, on change la résolution par S4. Enfin, il faut changer l'image PNG en image JPG. On utilise la recherche par avant pour chercher des chaînes qui changent l'image JPG en image PNG comme l'exemple précédent. Pour changer l'image PNG en JPG, on utilise la recherche par après. En général, la recherche par après et la recherche par avant sont similaires. La recherche par après part d'un service dans l'ensemble des services potentiels, et finit si on trouve un service dont la sortie est le document adapté. Après la recherche par avant et la recherche par après, on obtient des chaînes qui résolvent un problème. Le diagramme dans la figure 3.7 montre le processus de recherche des services d'adaptation dans ce cas :

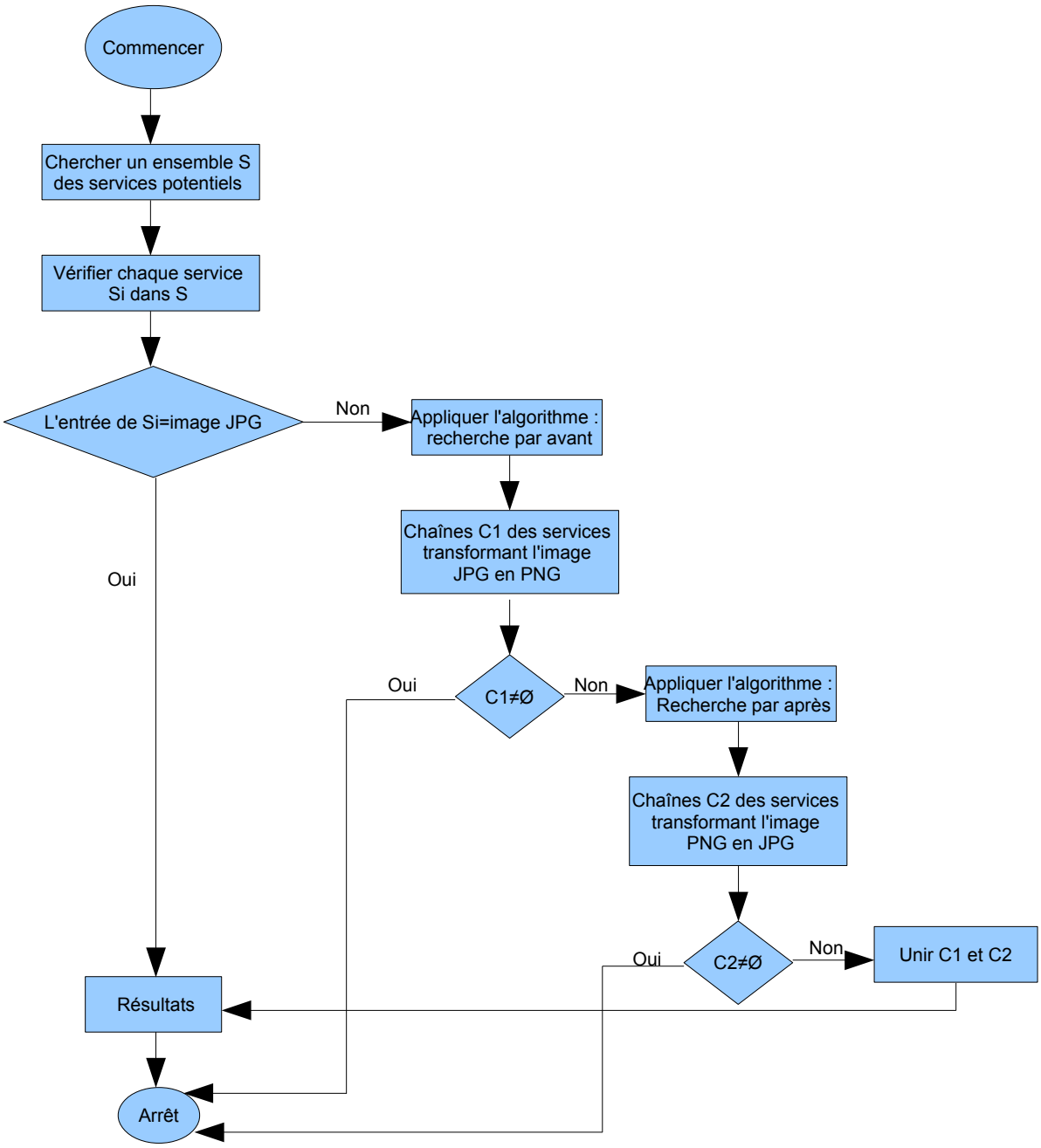

Figure 3.7 : Exemple d'un processus de recherche des services d'adaptation

On peut utiliser seulement la recherche par avant ou bien les deux modes de recherche, cela dépend du scénario d'adaptation. Pour expliquer clairement, je donne cinq scénarios pour discuter :

- Le premier scénario est le plus simple. On a seulement un problème de conflit de format, et on utilise seulement la recherche par avant pour trouver toues les chaînes résolvant le

problème.

- Le deuxième scénario est plus complexe. On a un problème de conflit de la résolution, mais il faut utiliser les deux algorithmes de recherche.
- Le troisième scénario, on commence à examiner le cas où il y a plus d'un problème. Dans ce troisième scénario, on aura les problèmes de conflits de format et de résolution. Il nécessite seulement la recherche par avant.
- Dans le quatrième scénario, l'utilisateur indique les formats de document qu'il accepte. Par conséquent, il faut coordonner les deux algorithmes de recherche.
- Dans le cinquième scénario, on a des conflits comme ceux dans le quatrième scénario, mais l'utilisateur n'indique pas le format de document multimédia.

Les scénarios discutés représentent tous des cas d'utilisation du processus de recherche de chaînes de services d'adaptation.

Le premier scénario : l'utilisateur demande une image en format JPEG. L'image sur le serveur multimédia est en format de PNG. Le serveur multimédia change donc le format de PNG à JPEG. La liste de services d'adaptation du serveur multimédia est donnée dans la table ci-dessous :

Nom de service	Description
S1	Convertir l'image de PNG en JPEG
S2	Convertir l'image de PNG en GIF
S3	Convertir l'image de GIF en TIFF
S4	Convertir l'image de TIFF en JPEG

Table 3.9 : Liste de services du premier scénario

À partir de la table, on trouve que le service S1 et la chaîne S2→S3→S4 satisfont le demande de l'utilisateur. Dans ce cas, l'idée est la recherche par avant. À partir du service A qui produit l'image au format JPEG, on cherche le service B dont la sortie est l'entrée du service A. On applique de la même manière un nouveau service jusqu'à ce que l'entrée du nouveau service soit l'image en format PNG. Concrètement, on part du service S4 parce qu'il produit l'image en format JPEG. Son entrée est l'image en format TIFF. Le service qui a la sortie produisant une image au format TIFF est S3. L'entrée de S3 est l'image au format GIF. Le service dont la sortie est l'image au format GIF est S2. L'entrée de S2 et le format de l'image originale sont les mêmes. L'algorithme est donc terminé. On a la chaîne

S2→S3→S4.

Le deuxième scénario : l'utilisateur demande une image en format JPEG avec la résolution plus petite que celle de l'image sur le serveur multimédia. L'image source est en format JPEG. Le serveur change donc la résolution de l'image. La liste de services d'adaptation du serveur multimédia est donnée dans la table ci-dessous :

Nom de service	Description
S1	Changer la résolution de l'image GIF. L'entrée est l'image en format de GIF. La sortie est l'image en format de GIF.
S2	Convertir l'image de JPEG en GIF
S3	Convertir l'image de GIF en TIFF
S4	Convertir l'image de TIFF en JPEG

Table 3.10 : Liste de services du deuxième scénario

En regardant la table, on trouve qu'il n'y a aucun service qui modifie directement la résolution de l'image en format JPEG. Pour changer la résolution, il faut utiliser le service S1. Mais S1 est appliqué seulement pour une image au format GIF. Donc il faut modifier également le format de la ressource image en GIF. Comme l'utilisateur demande une image en format JPEG, après l'exécution de S1, il faut convertir l'image GIF en JPEG. On trouve alors la chaîne : S2→S1→S3→S4. Pour trouver la chaîne, on applique la recherche par avant et la recherche par après . La recherche par avant correspond au premier scénario. La recherche par après est l'inverse de la recherche par avant. À partir du service A, on cherche le service B dont l'entrée est la sortie du service A. On applique de la même manière au nouveau service jusqu'à ce que la sortie du nouveau service soit l'image en format JPEG. Concrètement sur l'exemple, on trouve le service S1 changeant la résolution. À partir du service S1, on applique la recherche par avant : la sortie de S2 et l'entrée de S1 sont identiques. L'entrée de S2 est l'image en format JPEG, alors la recherche par avant est terminée. Après la recherche par avant, on a la chaîne S2→S1. Ensuite, on réalise la recherche par après à partir de S1: la sortie de S1 est l'entrée de S3, alors on obtient S3. La sortie de S3 est l'entrée de S4, alors on obtient S4. La sortie de S4 est l'image en format JPEG, alors la recherche par après est terminée. On a la chaîne S1→S3→S4. Après les recherches, on a la chaîne S2→S1→S3→S4.

Le troisième scénario : l'utilisateur demande une image en format de JPEG avec la résolution plus petite que celle de l'image sur le serveur multimédia. L'image de ressource est en format PNG. Le serveur multimédia change le format de PNG à JPEG et change la résolution de

ressource. La liste de services d'adaptation est donnée dans la table 3.11 :

Nom de service	Description
S1	Changer la résolution de l'image GIF. L'entrée est l'image en format de GIF. La sortie est l'image en format de GIF.
S2	Convertir l'image de PNG en GIF.
S3	Convertir l'image de GIF en JPEG.

Table 3.11 :Liste de services du troisième scénario

Dans ce cas, tout d'abord on cherche la chaîne qui résout le problème de changement de format comme le premier et le deuxième scénarios. On obtient la chaîne S2→S3. Ensuite, on cherche le service qui résout le problème de changement de résolution, c'est le service S1. Pour ajouter S1 à la chaîne S2→S3, on cherche la position convenable dans la chaîne. Si on ne trouve aucune la position possible, c'est-a-dire qu'on ne peut pas ajouter le service S1 à la chaîne S2→S3. Cela signifie qu'il n'y a aucun service qui résout tous les problèmes. Dans ce cas, on ajoute le service S1 à la position après S2, c'est-à-dire on a la chaîne S2→S1→S3.

Le quatrième scénario : l'utilisateur demande une image au format JPEG avec une résolution et une taille mémoire plus petites que celles de l'image stockée sur le serveur multimédia. La ressource image est au format JPEG. Le serveur multimédia change la résolution et la taille mémoire de la ressource. La liste de services d'adaptation est donnée dans la table 3.12 :

Nom de service	Description
S1	Changer la résolution de l'image GIF. L'entrée est l'image en format de GIF. La sortie est l'image en format de GIF.
S2	Convertir l'image de JPEG en GIF.
S3	Convertir l'image de GIF en JPEG.
S4	Changer la taille de mémoire. L'entrée est l'image en format de JPEG. La sortie est l'image en format de JPEG.

Table 3.12 : Liste de services du quatrième scénario

Dans le troisième scénario, tout d'abord, on vérifiait le problème de changement de format, et puis, on ajoutait les autres services qui résolvent les autres problèmes aux chaînes trouvées. Mais dans ce quatrième scénario, les problèmes ne concernent pas le changement de format. Dans ce cas, on choisit un problème, le problème de modification de la taille de

mémoire par exemple. On cherche les chaînes qui résolvent ce problème comme dans le deuxième scénario. C'est la chaîne qui a le service S4. Ensuite, on cherche la position dans la chaîne pour ajouter le service qui résout le problème de changement de la résolution : S1. Si la position n'existe pas, on cherche les chemins de S1 à S4 par la recherche par avant. Dans ce cas, on a la chaîne S2→S1 ce qui donne comme chaîne finale S4→S2→S1. Comme la sortie de S1 n'est pas l'image en format de JPEG, alors on cherche les chaînes qui changent le format GIF en format JPEG. Dans ce cas, on obtient une chaîne ayant un élément S3. Enfin, on obtient la chaîne : S4→S2→S1→S3.

Le cinquième scénario : l'utilisateur demande une image avec la résolution et la taille de mémoire plus petites que celles de l'image sur le serveur multimédia. La ressource image est au format JPEG. Le serveur multimédia change la résolution de ressource ce qui change son empreinte mémoire. La liste des services d'adaptation est donnée dans la table 3.13 :

Nom de service	Description
S1	Changer la résolution de l'image GIF. L'entrée est l'image en format de GIF. La sortie est l'image en format de GIF.
S2	Convertir l'image de JPEG en GIF
S3	Convertir l'image de PNG en GIF
S4	Changer la taille de mémoire. L'entrée est l'image en format de PNG. La sortie est l'image en format de PNG.

Table 3.13 : Liste de services du cinquième scénario

Dans ce cas-ci, l'utilisateur accepte n'importe quel format. Dans ce cas, on choisi un problème, le problème de changement de la résolution par exemple. On applique la recherche par avant, on obtient la chaîne : S2→S1 qui résout ce problème. Avec le problème de changement de la taille de mémoire, on trouve que S4 change la taille de mémoire. On cherche dans la chaîne la position pour ajouter S4. Si la position n'existe pas, on cherche la route de S4 à la fin de la chaîne, c'est le service S1, par la recherche par avant. Dans ce cas, on trouve la chaîne: S1→S4. Par unification des résultats on obtient la chaîne: S2→S1→S4.

3.3.7.1. Les étapes de recherche par avant et recherche par après

Deux algorithmes : recherche par avant et recherche par après sont importants dans l'algorithme pour trouver les services d'adaptation. Je construis l'ensemble des services en arbre. La recherche par avant et la recherche par après deviennent des recherches sur l'arbre.

a) recherche par avant : On suppose qu'on fait la recherche à partir du service S_1. Comme je l'explique dans la partie ci-dessus, l'idée de la recherche par avant est que la sortie du service

précédent est l'entrée de service suivant. On construit un arbre où l'entrée du nœud fils est la sortie de nœud parent. Le processus de recherche par avant suit les étapes suivantes :

- Première étape : on cherche un service dans l'ensemble des services disponibles dont la sortie est l'entrée de S_1. En plus le service n'est pas encore visité, ou il est visité mais son fils n'est pas S_1. On l'appelle S_{21}. On applique ce processus à S_{21} pour trouver S_{31}. Après le processus, on a l'arbre suivant :

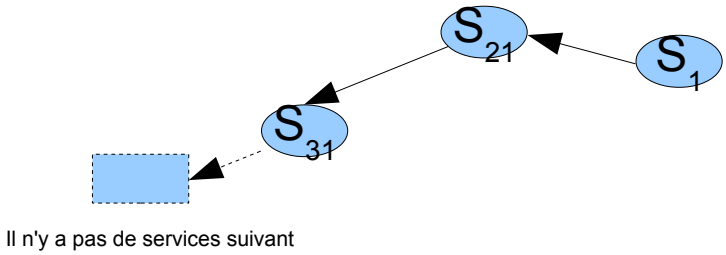

Figure 3.8 : Première étape de la recherche par avant

Dans la figure ci-dessus, on trouve que S_{31} ne connecte rien parce qu'il n'a pas de parent, ou il a des parents mais tous ces parents sont déjà visité par S_{31}. Dans le cas où le nœud n'a pas des parents qui ne sont pas encore visités, on va dans le sens inverse, et le nœud enfant, c'est S_{21}. On continue à appliquer le processus de recherche à S_{21}. Ce processus est réalisé jusqu'à qu'on trouve un service dont l'entrée est le format de ressource, c'est le service S_5, par exemple. On a une chaîne de service adapté : $S_1 \rightarrow S_{21} \rightarrow S_{32} \rightarrow S_5$ comme la figure 6.3b. On ajoute la chaîne dans une liste de chaîne.

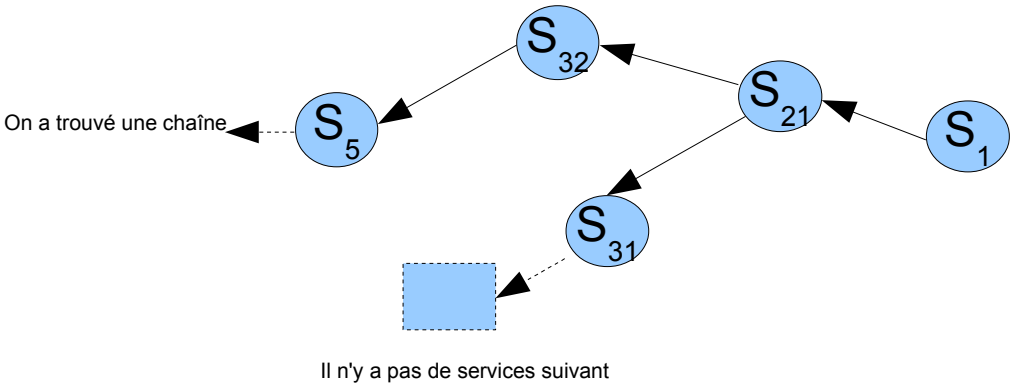

Figure 3.9 : Première étape de la recherche par avant

- Deuxième étape : Quand on trouve le service dont l'entrée est le format de la ressource, on va à l'inverse de son enfant, dans ce cas, c'est S_{32}. Les parents de S_{32} sont visités, on va à S_{21}. On applique le processus comme celui dans première étape, on a une autre chaîne : $S_1 \rightarrow S_{21} \rightarrow S_{33} \rightarrow S_{41} \rightarrow S_5$. L'arbre est suivant :

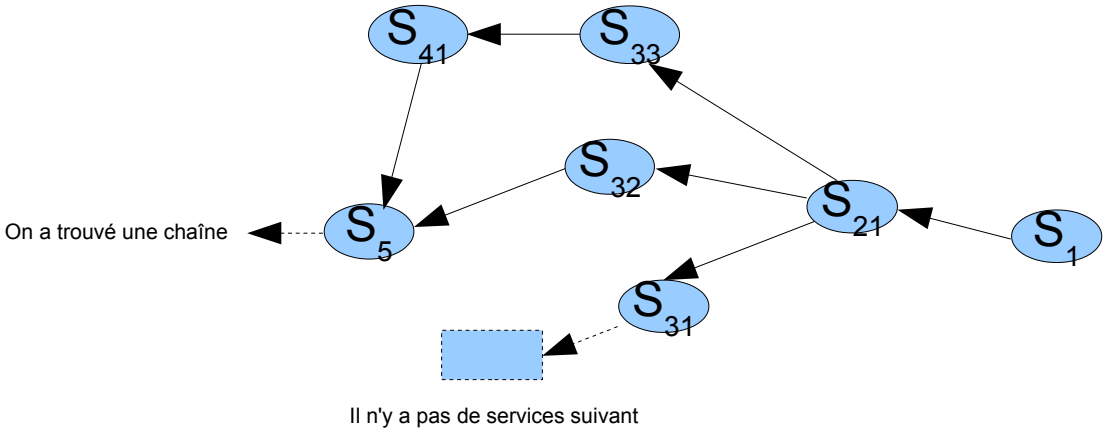

Figure 3.10 :Deuxième étape de la recherche par avant

Quand on trouve une nouvelle chaîne, on va à l'inverse, c'est S_{41}. S_{41} n'a pas de parents qui ne sont pas visités, alors on continue à aller à l'inverse, c'est S_{33}. On continue à aller à S_{21}. Tous les parents de S_{21} sont visités, on va alors à S_1. On suppose que S_{22} est un autre parent de S_1. On va à S_{22}. S_{41} est un parent de S_{22}. Donc on continue à aller à S_{41}. À ce moment, S_{41} n'a aucun parent qui n'est pas encore visité. On perd donc la chaîne $S_1 \rightarrow S_{22} \rightarrow S_{41} \rightarrow S_5$. Pour résoudre ce problème, quand un nœud n'a aucun parent qui n'est pas encore visité, on configure non-visitation pour les parents de ce nœud. Selon cette solution, quand on part de S_5 à S_{41}, le nœud S_5 n'est pas invité parce que S_{41} n'a aucun parents qui ne sont pas encore visités. Donc quand on part de S_{22} à S_{41}, S_{41} a un parent, c'est S_5. On a une nouvelle chaîne $S_1 \rightarrow S_{22} \rightarrow S_{41} \rightarrow S_5$. L'arbre est suivant :

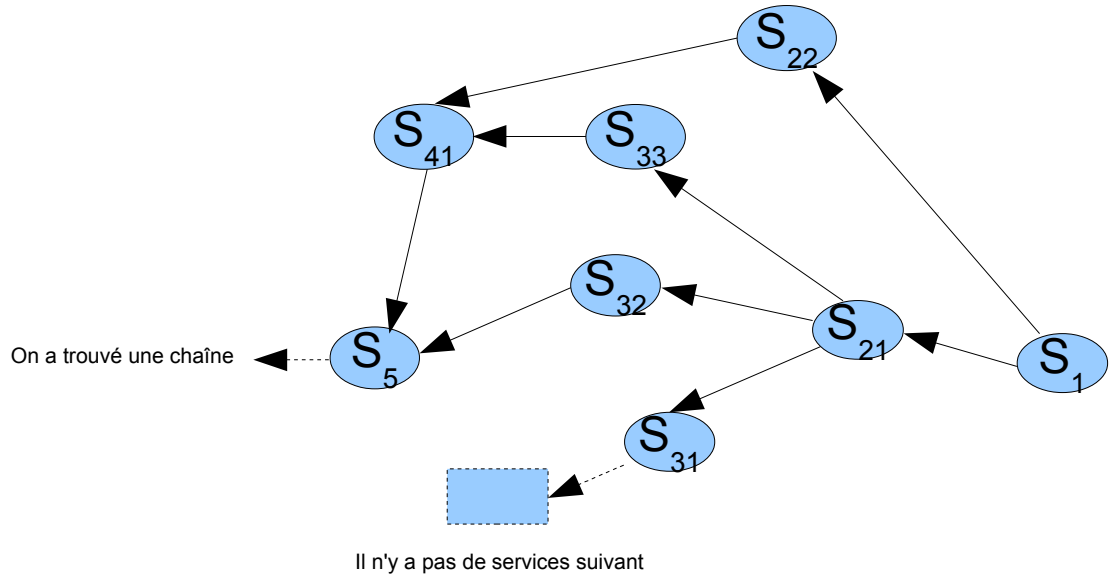

Figure 3.11 : Deuxième étape de la recherche par avant

- Troisième étape : on trouve que les services dans toutes les chaînes trouvées dans les

deux étapes ci-dessus, ne sont pas ordonnés. Dans toutes les chaînes, S_5 est en dernière position, S_1 est en première position. En effet, il faut exécuter premièrement le service S_5, parce que son entrée est le média source. Le service S_1 est exécuté dernièrement. Donc il faut les déposer en ordre. Après cette troisième étape, les chaînes de services d'adaptation sont : $S_5 \rightarrow S_{32} \rightarrow S_{21} \rightarrow S_1$, $S_5 \rightarrow S_{41} \rightarrow S_{22} \rightarrow S_1$

Dans le processus de recherche, on s'intéresse au format de sortie et d'entrée. Donc les services dans une chaîne de résultats ont peut-être des types d'action différents. Par exemple, S_1 transforme l'image GIF en image JPEG; S_{21} transforme la vidéo en format de MOV en image GIF; S_{22} transforme la vidéo 3GP en image GIF ; S_{32} transforme l'image GIF en vidéo en format de MOV; S_{33} transforme la vidéo 3GP en vidéo MOV ; S_{41} transforme l'image GIF en vidéo 3GP ; S_5 transforme l'image PNG en image GIF. Selon les fonctions des services, on trouve que le type d'action de S_1, S_{33}, S_5 sont «*transcodage* », celle de S_{21},S_{22} , S_{32}, S_{41} sont «*transmodage* ». Le résultat est montré dans le figure ci-dessous :

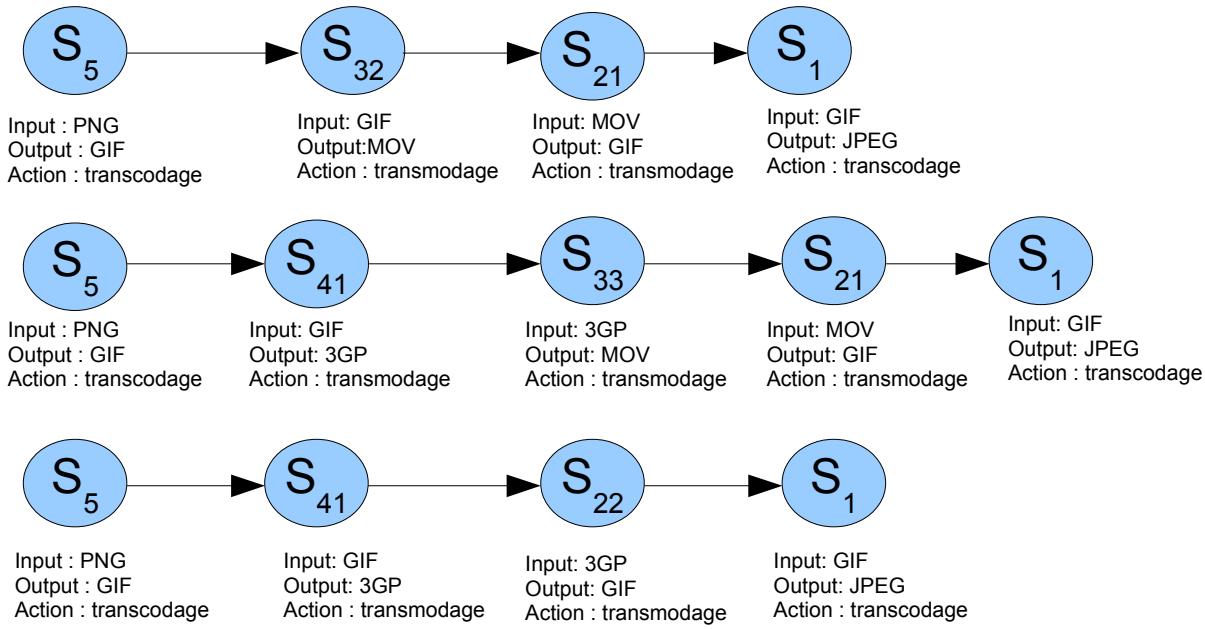

Figure 3.12 : Le résultat de la recherche par avant

b) recherche par après

La recherche par après est implémentée comme la recherche par avant, mais on construit l'arbre selon la règle : la sortie de nœud parent est l'entrée de nœud enfant. Après les deux premières étapes, des services dans une chaîne sont déposés en ordre. Donc on n'a pas besoin de faire la troisième étape. Un exemple est montré dans le figure au-dessous :

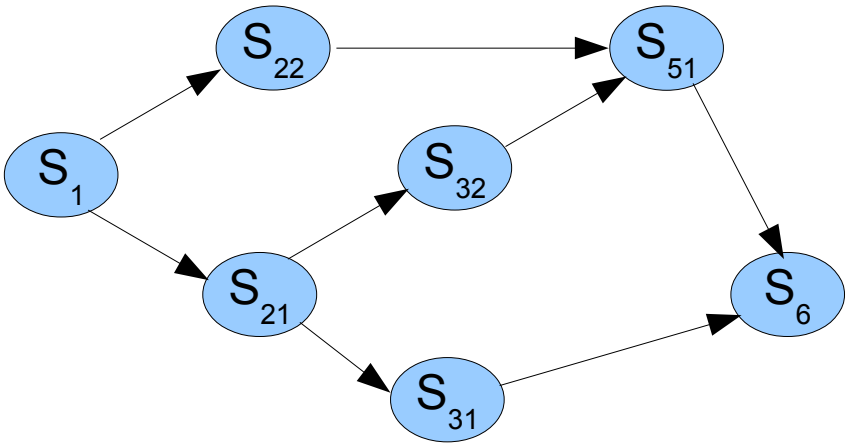

Figure 3.13 : Arbre de recherche de la recherche par après

Dans la recherche par après, les services dans une chaîne de résultats ont peut-être des types d'action différents. Les chaînes de résultats dans cet exemple sont illustrées ci-après :

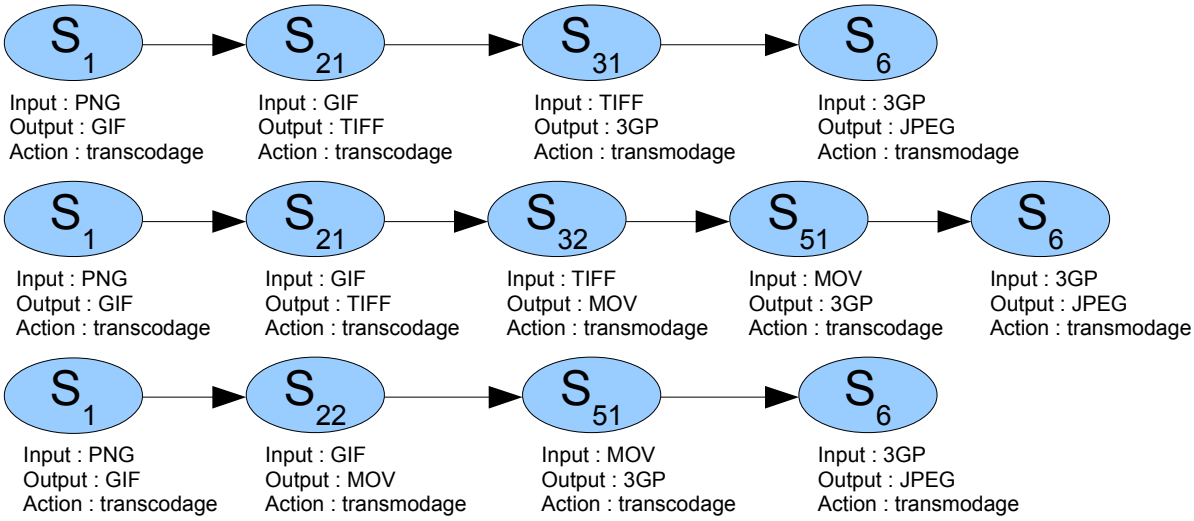

Figure 3.14 : Le résultat de la recherche par après

3.3.7.2. Optimisation de la recherche par avant et la recherche par après

Dans la recherche par avant et après, quand on va à un nœud, on cherche le prochain nœud en vérifiant chaque service disponible. Si ce service satisfait les règles de construction de l'arbre, il est le nœud prochain. Si la quantité de services est très grande, à chaque nœud, on doit faire beaucoup de comparaison entre les autres services et la règle de construction de l'arbre. Par conséquent, le système peut être ralenti. Ceci peut être gênant surtout dans une situation de mobilité de l'utilisateur, c'est-a-dire l'adaptation doit être réactive et fournir une

solution rapidement. C'est une raison pour laquelle, la performance du système est réduite. Pour surmonter le problème, pour chaque service, on établit des relations entre ce service et les autres. Si la sortie d'un service est l'entrée de l'autre, ces services sont en relation. Par exemple : à supposer qu'on dispose de 4 services : S_1 (input = JPEG, output = PNG) ; S_2 (input = JPEG, output = 3GP) ; S_3 (input = PNG, output = GIF); S_4 (input = 3GP, output = JPEG). Le résultat est montré dans le figure ci-dessous :

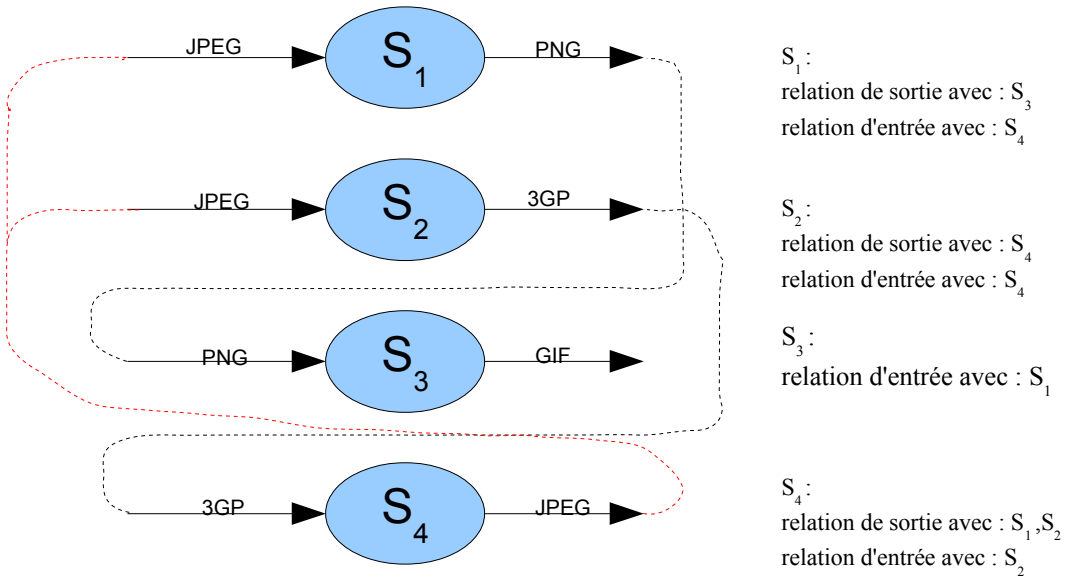

Figure 3.15: Relation entre des services

Basé sur les relations d'un service, on cherche facilement le prochain nœud. Par exemple, dans la recherche par avant, quand on part à S_1, on trouve dans les relations d'entrée, on obtient le service S_4. Si S_4 n'est pas encore invité, il est le prochain nœud.

Dans cette partie, j'ai expliqué l'optimisation qui permet de chercher rapidement le nœud suivant. Dans la recherche par avant et la recherche par après, la quantité de nœuds est une raison qui influence la vitesse de recherche. Si la quantité de nœud est importante, la vitesse de recherche est réduite parce qu'on ne visite pas touts les nœuds dans l'arbre de recherche. On réduit ainsi la quantité de nœud.

Comme on utilise la recherche par avant et la recherche par après pour traiter le problème de format c'est-a-dire on recherche les chaînes qui changent un format en un autre. Les chaînes contiennent les services qui changent le format du document multimédia. Donc dans la recherche des chaînes, on vérifie seulement les services changeant le format de document multimédia. C'est la raison pour laquelle, quand on établit la relation entre des services, on s'intéresse seulement aux services qui changent le format de document multimédia. Donc, dans l'arbre de recherche de services , il n'y a pas de services qui ne modifient pas le format

de document multimédia. Avec un grand nombre de services, l'élimination des services qui ne changent pas le format de document multimédia améliore la vitesse de recherche. Par exemple : on exécute la recherche par avant à partir d'un service S_1 (ou nœud S_1). Selon la relation d'entrée, S_1 a N_1 nœuds voisins qui sont les services changeant le format de document multimédia. Les nœuds voisins de S_1 ont N_2 nœuds voisins. À supposer qu'on construit un arbre avec $T = N_1 + N_2 + ... + N_k$ nœuds. On visite successivement tous les T nœuds. Maintenant, on suppose que S_1 a M nœuds voisins qui ne changent pas le format. On vérifie un nœud dans ces M nœuds, appelé S_{12}. Comme S_{12} ne change pas le format, alors il a N_1 nœuds voisins qui sont eux-mêmes N_1 voisins de S_1. Si on ne supprime pas S_{12}, avec S_{12}, il faut visiter T+1 nœuds. Avec M nœuds voisins de S_1 qui ne changent pas le format, on doit visiter M(T + 1) nœuds. Donc le total des nœuds dans l'arbre est égal à T + M(T+1) . Si M et T sont très grands, alors T + M(T+1) >>T. Donc on perd beaucoup de temps pour rechercher toutes les chaînes de services. Dans les cas où il y a le problème de changement de format et des problèmes qui ne changent pas le format (changement de résolution, de la taille de mémoire,...) comme le troisième scénario, tout d'abord, on résout le problème de changement de format pour obtenir des chaînes de services. Ensuite, on cherche des positions pour ajouter des services qui résolvent les problèmes qui ne changent pas le format.

3.3.7.3. L'algorithme de recherche

Je donne les étapes pour implémenter l'algorithme de recherche :

1. Trouver le problème de changement de format dans la liste de problèmes. S'il existe, passer à l'étape 2. Si non, aller à l'étape 6

2. Trouver les services dont la sortie est le document avec le format adapté. S'ils existent, passer à l'étape 3. Si non, l'algorithme est terminé.

3. Exécuter la recherche par avant à partir de chaque service qui est trouvé dans l'étape 2 pour chercher les chaînes qui changent le format de ressource en format adapté. S'il n'existe aucune chaîne, l'algorithme est terminé. Sinon, passer à l'étape 4.

4. Trouver les services qui résolvent les problèmes qui ne changent pas le format. S'il n'existe aucun service, l'algorithme est terminé. Sinon, passer à l'étape 5.

5. Ajouter les services trouvés dans l'étape 4 aux chaînes trouvées dans l'étape 3. S'il existe une chaîne qui ne résout pas tous les problèmes dans la liste de problèmes, supprimer la chaîne.

6. Vérifier le premier problème dans la liste des problèmes. Trouver les services

correspondant au problème.

7. Exécuter la recherche par avant à partir de chaque service qui est trouvé dans l'étape 6 pour chercher les chaînes qui changent le format de le ressource au format d'entrée du service. S'il n'existe aucune chaîne, l'algorithme est terminé.

8. Si l'utilisateur ne demande pas le format, aller à l'étape 11.Sinon,passer à l'étape 9 .

9. Exécuter la recherche par après à partir de chaque service trouvé dans l'étape 6 pour chercher les chaînes qui change le format de la sortie du service en format de ressource. S'il n'existe aucune chaîne, l'algorithme est terminé.

10. Unir les chaînes trouvées dans les étapes 7 et 9.

11. Trouver les services correspondant aux autres problèmes dans la liste des problèmes. Si l'utilisateur ne demande pas le format, aller à l'étape 13. Sinon, passer à l'étape 12.

12. Ajouter les services aux chaînes trouvées dans l'étape 10. S'il existe une chaîne qui ne résout pas toutes les problèmes dans la liste des problèmes, supprimer la chaîne.

13. Ajouter les services trouvés dans l'étape 11 aux chaînes trouvées dans l'étape 6. S'il existe un service qu'il n'est pas possible d'ajouter, changer l'étape 14. Sinon, l'algorithme est terminé.

14. Exécuter la recherche par après à partir du service qu'il n'est pas possible d'ajouter aux chaînes trouvées dans l'étape 6.

15. Unir les chaînes trouvées dans les étapes 6 et 14. L'algorithme est terminé.

Les étapes de l'algorithme de recherche est présenté dans le diagramme dans le figure :

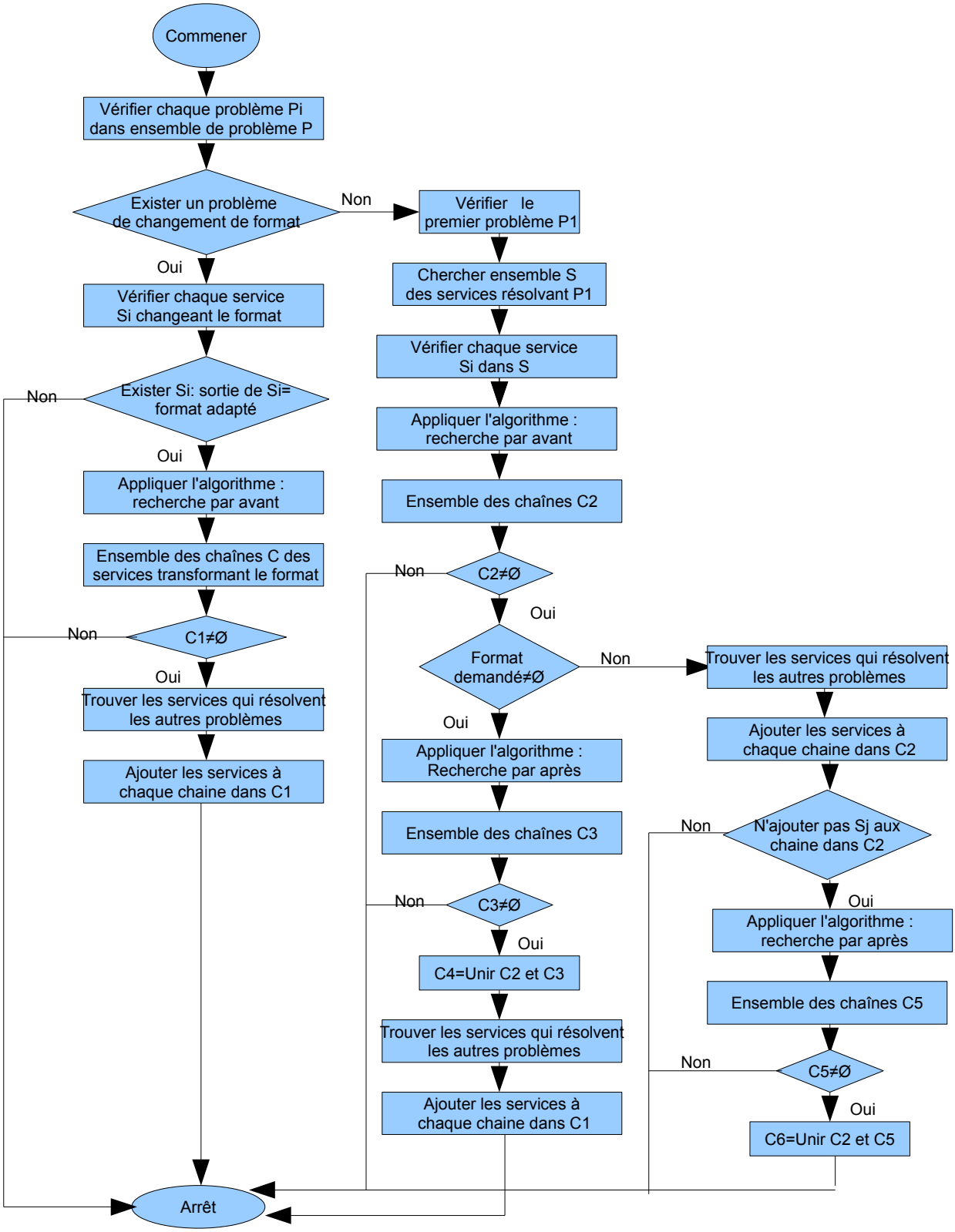

Figure 3.16 : Algorithme de recherche de services d'adaptation

En regardant les étapes de l'algorithme de recherche, on trouve qu'on peut obtenir tous les résultats (ou toutes les chaînes) qui résolvent tous les problèmes d'adaptation. Donc on peut choisir la meilleure chaîne qui a le plus petit poids pour exécuter l'adaptation.

La complexité de l'algorithme de recherche par avant et celle de l'algorithme de recherche par après sont identiques, et sont égales à O(N) où N est le nombre de services changeant le format de document multimédia.

Les performances(par exemple, la vitesse d'exécution) de l'algorithme de recherche par après sont égales à celles de l'algorithme de recherche par avant.

Un inconvénient de l'algorithme de recherche est qu'il est difficile de l'implémenter parce que dans chaque étape, il faut vérifier beaucoup de conditions.

3.3.8. Description de l'algorithme de recherche

Dans cette partie, je présente plus formellement l'algorithme de recherche. Je donne certaines définitions qui seront utilisées dans la description de l'algorithme.

Définition 1 : la valeur d'une caractéristique V est une série de mots.

Par exemple V = {".PNG"} représente la valeur de caractéristique de format.

Définition 2 : un ensemble de valeurs d'une caractéristique est un ensemble de V. Il est représenté comme PV.

Par exemple : PV = $\{V_1, V_2, V_3\}$.

Définition 3 : une caractéristique dans un profil est défini par des propriétés : le nom de caractéristique et son ensemble des valeurs. Il est représente comme D (N, PV) où :

- N: le nom d'une caractéristique
- PV : l'ensemble des valeurs de la caractéristique

Définition 4 : une caractéristique de ressource est définit par des propriétés : le nom de caractéristique et sa valeur . Il est représente comme R (N, V) où :

- N: le nom de la caractéristique
- V: la valeur de la caractéristique

Définition 5: un problème est une différence entre un caractéristique de ressource et celui du profil. Un problème existe si:

$$(N.D = N.R) \wedge (V.R \notin PV.D)$$

Il est défini par les termes : la valeur de ressource, l'ensemble des valeurs du caractéristique de profil, le nom de caractéristique, le type de ressource, le type d'action,le format que l'utilisateur accepte, le format de ressource, et une booléen de la vérification. Il est

représenté comme : Prb(V, PV, N,F,M, A, FP, FR, EXE) où :

- V: la valeur de la caractéristique de la ressource initiale.
- PV : l'ensemble des valeurs de la caractéristique du profil
- N: le nom de la caractéristique
- M: le type de la ressource
- A: l'action du service
- FP: l'ensemble des formats que l'utilisateur accepte. $FP \in PV.D(''format'', PV)$
- FR : le format de la ressource
- EXE: la vérification

Définition 6: un ensemble des problèmes PCF = {Prb_1, Prb_2,..., Prb_m } est l'ensemble de problèmes de changement de format si

$A.Prb_i$ = « transcodage »

ou $A.Prb_i$ = « transmodage » i = 1,m

Définition 7 : un service d'adaptation est défini par les termes : le nom de service,son entrée, sa sortie, le type de la ressource que l'on peut modifier, le type d'action et une booléen de vérification. Il est représenté comme : S(N, I, O, M, A, EXE) où :

- N: le nom de service
- I: entrée du service
- O: sortie du service
- M: le type de la ressource
- A: action du service
- EXE: la vérification

Définition 8: un service est appelé NoeudCheck s'il vérifie un autre. Il représente comme NoeudCheck(SV) où SV est le service vérifié.

Définition 9 : un ensemble de services Avant(S) = { S_1, S_2,..., S_p} est appelé l'ensemble avant de S si

$O.S_i = I.S$ i = 1,p

Définition 10 : un ensemble de services Apres(S) = { S_1, S_2,..., S_p} est rappelé l'ensemble après de S si

$I.S_i = O.S$ i = 1,p

Définition 11: un ensemble de services SCF = { S_1, S_2,..., S_n } est appelé l'ensemble de services changeant le format si

$A.S_i$ = « transcodage »

ou $A.S_i$ = « transmodage » i = 1,n

Définition 12: une chaîne de services C est un ensemble de services où les services sont disposés en ordre.

Théorème 1 : On appelle une chaîne C est une chemin de S_a à S_z si C est de forme :

$C = \{ S_a, S_1, ..., S_z\}$ où $O.S_i = I.S_{i+1}$

Supposons deux services S1, S2

Si : L1 = {C / C est une chemin de S1 à Sz et $S2 \in C$ }

L2 = {C / C est une chemin de S2 à Sz}

Alors : chaque $C2 \in L2$, il existe $C1 \in L1$, $C2 \subset C1$

Preuve :

Comme dans L1 : $S2 \in C$, alors il existe une route de S1 à S2, c'est-a-dire il existe une chaîne C12 = {S1, …., S2}.

Obtenir n'importe quelle chaîne dans L2, C2z = {S2,....,Sz}. Donc

C1 = C12 U C2z = {S1,....S2,....Sz} est une route de S1 à Sz. Alors $C1 \in L1$.

Comme $C2z \subset C1$, alors le théorème est prouvé □

Quand on vérifie un premier problème, on a les chaînes qui résolvent ce problème. On ajoute des services à ces chaînes pour obtenir les résultats finaux. Basé sur le théorème, quand on applique le processus au deuxième problème, on obtient les résultats finaux qui sont identiques au premier problème. Donc, dans le cas où il y a beaucoup de problèmes comme dans le quatrième scénario et le cinquième scénario, on vérifie seulement un problème pour obtenir tous les résultats qui résolvent tous les problèmes. Dans les étapes d'implémentation de l'algorithme de recherche qui est donné dans la partie 5.7.3, je vérifie le premier problème.

L'algorithme de recherche consiste en deux algorithmes : l'algorithme de recherche par avant et l'algorithme de recherche par après. Ces algorithmes sont représentés en détail dans l'annexe.

Chapitre 4 : Implémentation et évaluation

4.1. Implémentation du prototype

Comme expliqué dans le chapitre 3, dans le modèle général, on a un serveur d'annuaire et des serveurs multimédias. Chaque serveur multimédia a un fichier de description des services représentant les services que le serveur peut exécuter. Quand un serveur multimédia est actif, il envoie son fichier de description au serveur d'annuaire. L'utilisateur souhaite un document multimédia satisfaisant toutes les contraintes décrites dans le profil(langage par exemple). Il envoie un fichier profil.xml au serveur multimédia qui enregistre la ressource. Le serveur multimédia exécute l'algorithme de recherche pour chercher les services qui vont permettre l'adaptation du document. On a deux cas dans le processus de recherche:

1. Le serveur multimédia trouve un service ou les chaînes de services qui satisfont les demandes de l'utilisateur. Il exécute successivement ces services et enfin, il revoie au client le document adapté.
2. Le serveur multimédia ne trouve aucune chaîne de service, il demande au serveur d'annuaire de rechercher des chaînes de services afin d'envoyer les problèmes au serveur d'annuaire. Le serveur d'annuaire reçoit les problèmes, puis, il exécute l'extraction des descriptions de services. Basé sur ces descriptions de services ainsi que les problèmes, il applique l'algorithme de recherche pour trouver des chaînes de services. Le serveur d'annuaire cherche des chaînes de services et détermine quels serveurs exécutent les services dans les chaînes. Enfin, il envoie aux serveurs multimédias les messages qui représentent les informations pour exécuter les services. Les étapes d'implémentation du serveur d'annuaire sont montrées dans la figure 4.1 :

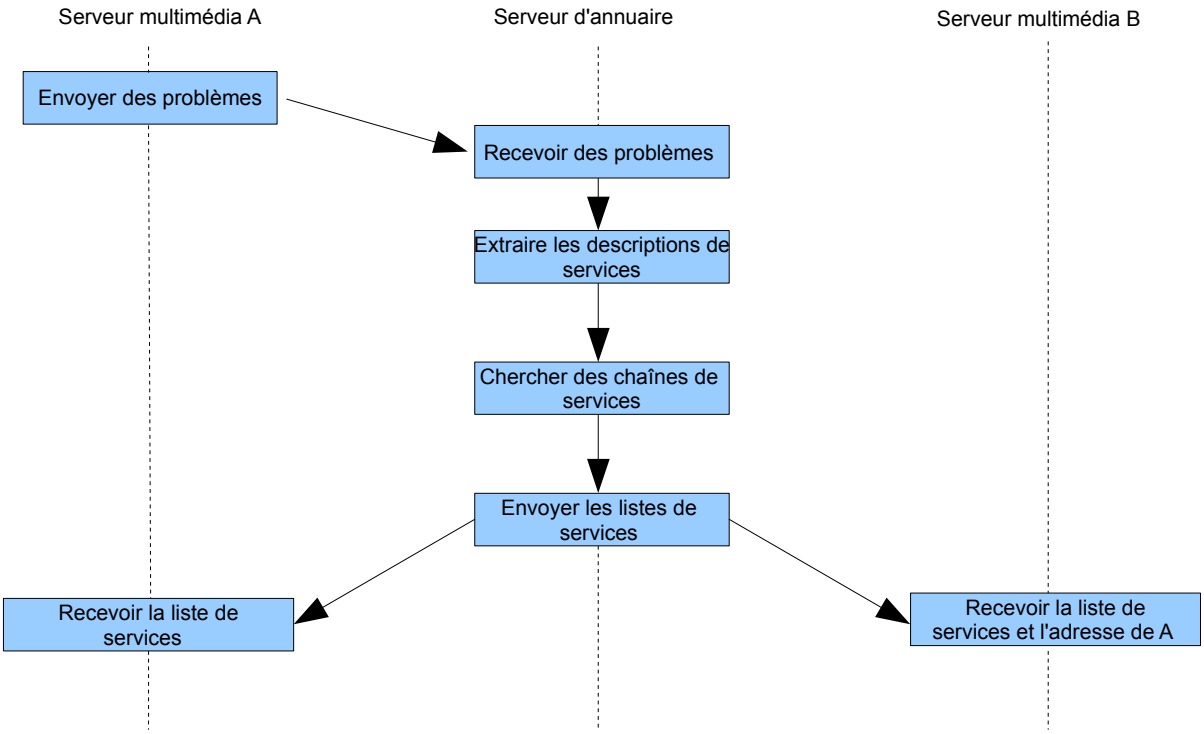

Figure 4.1 : Modèle général

Dans la figure, le serveur multimédia B doit se connecter au serveur multimédia A pour obtenir le document multimédia de A. Après avoir ce document multimédia, le serveur multimédia B exécute les services dans la liste de services. C'est la raison pour laquelle, le serveur d'annuaire envoie l'adresse IP du serveur A au serveur B.

Le serveur d'annuaire cherche les chaînes de services. S'il trouve les chaînes, il choisit une chaîne ayant le poids total minimum. Les services dans cette chaîne sont exécutés par des serveurs multimédias différents. On extrait les services correspondant à chaque serveur multimédia. Puis, on met les IDs des services en un message pour l'envoyer à son serveur multimédia. Le message que le serveur envoie à un serveur multimédia A est de la forme : [ID, Valeur], où ID est l'ID du service, Valeur correspond à la valeur dans le profil, c'est-à-dire la valeur d'adaptation utilisée dans le service. Je donne un exemple pour mieux expliquer mon propos. Supposons que le serveur d'annuaire trouve la meilleure chaîne : S1 →S2→S3→S4→S5 où les S1, S2, S4 sont exécutés sur le serveur multimédia A, et les services S3 et S5 sont exécutés sur le serveur multimédia B. Le message que le serveur d'annuaire revoie à A est de la forme :

MsgA = [ID_1,V_{11},V_{12},...,V_{1n},ID_2,V_{21},V_{22},...,V_{2m},0,ID_4, V_{41},....,V_{4l},0] où :

- ID_1, ID_2, ID_4 sont les identités des services S1, S2, S4 respectivement.

- $V_{11},V_{12},...,V_{1n}$ sont des valeurs dans le profil correspondant à S1; $V_{21},V_{22},...,V_{2m}$ sont des valeurs correspondant à S2 ; $V_{41},....,V_{4l}$ sont des valeurs correspondant à S4.
- 0: correspond à la position de S3, S5, c'est-à-dire S3, S5 n'existent pas dans la liste de services de A, on remplace S3, S5 par 0.

Le message correspondant au serveur multimédia B est le suivant:

$[0,0,ID_3,V_{31},V_{32},...,V_{3h},0,ID_5,V_{51},V_{52},...,V_{5g}]$.

Comme le serveur B doit se connecter au serveur A pour obtenir le document multimédia et pour renvoyer le document après exécution des services, alors le serveur d'annuaire envoie au serveur B l'adresse IP de A. Donc le message que B reçoit est :

$MsgB = [addrIP,0,0,ID_3,V_{31},V_{32},...,V_{3h},0,ID_5,V_{51},V_{52},...,V_{5g}]$

Le processus de changement des messages dans le système d'adaptation est présenté dans la figure 4.2 :

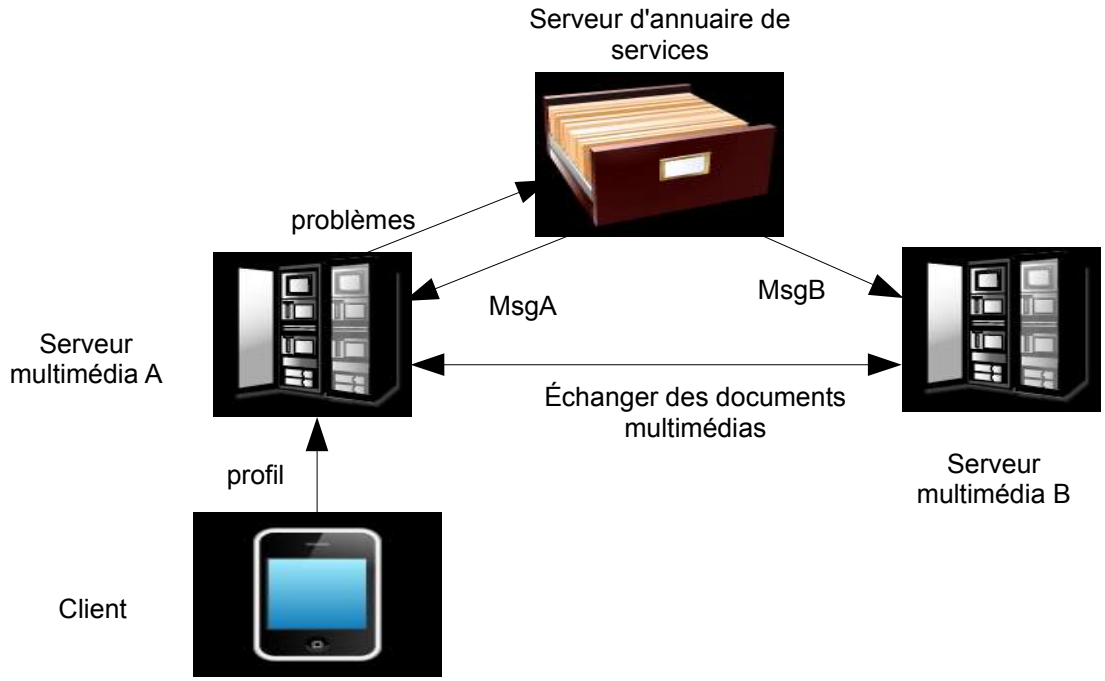

Figure 4.2 : Le processus de changement des messages

Lorsque le serveur A reçoit le message MsgA, on vérifie chaque élément du ce message selon la règle suivant :

- Si l'élément est égal à 0, le serveur A attend une demande d'autre serveur pour transférer le document multimédia.
- Si l'élément est égal à ID_i, le serveur A exécute le service S_i avec les valeurs $V_{i1},V_{i2},...,V_{ij}$.

Puis sur le serveur B, tout d'abord, on obtient l'adresse IP de serveur A. Ensuite, on traite le

message MsgB: si on a une série successif de 0, on remplace la série par 0. Donc, le message MsgB devient :MsgB = [0,ID_3,V_{31},V_{32},...,V_{3h},0,ID_5,V_{51},V_{52},...,V_{5g}]. Et puis, on vérifie chaque élément dans MsgB selon la règle suivant:

- Si l'élément est égal à 0, B se connecte à A pour changer le document multimédia.
- Si l'élément est égal à ID_i, le serveur B exécute le service S_i avec les valeurs V_{i1},V_{i2},...,V_{ij}.

Avec les messages MsgA et MsgB que j'ai donné, on a le schéma d'exécution comme illustré ci-après:

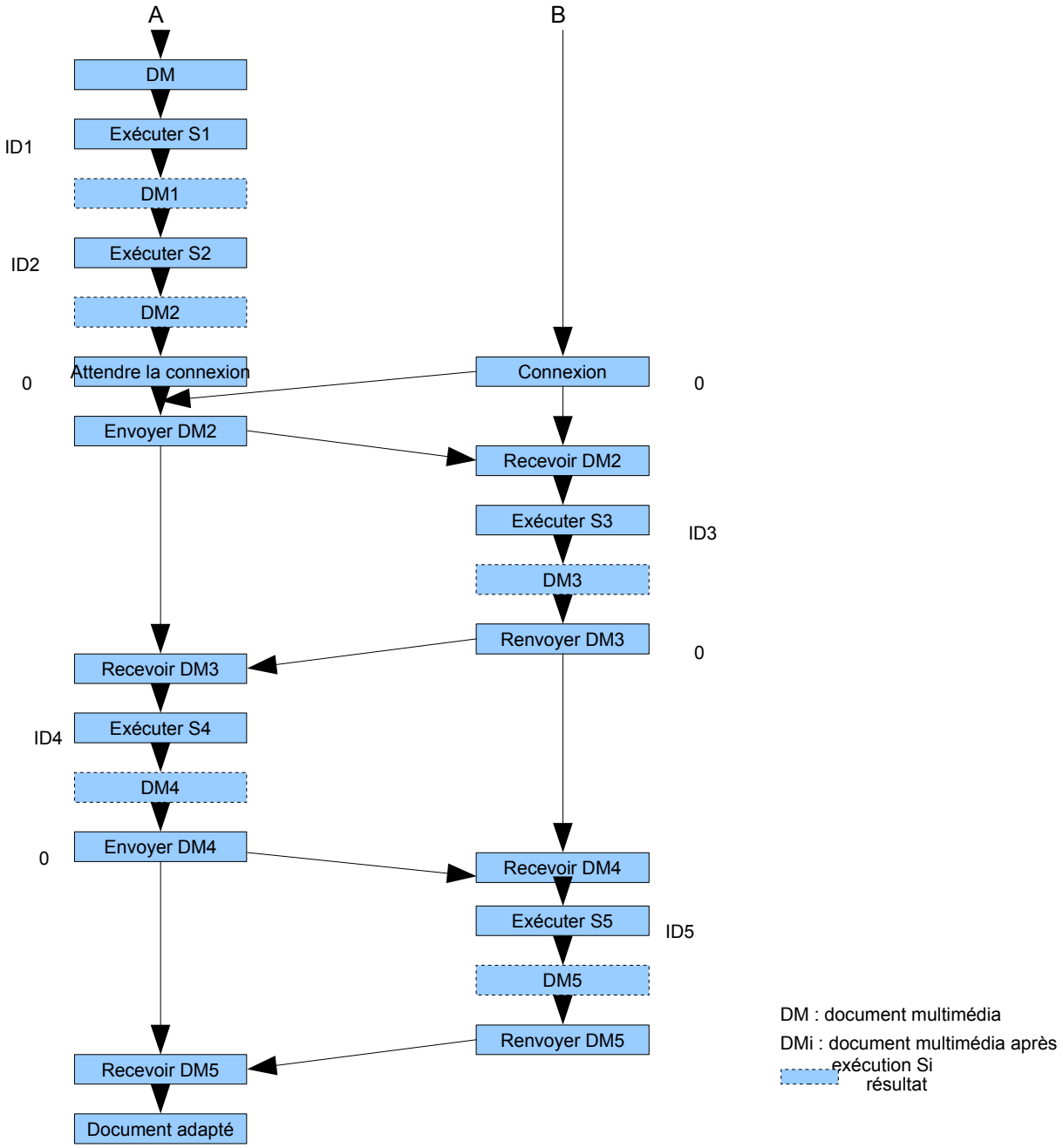

Figure 4.3 : Schéma d'exécution entre des serveurs multimédias

Après que tous les services dans la chaîne de services soient exécutés, le serveur multimédia A reçoit le document adapté que l'utilisateur accepte. Le serveur A renvoie au client ce document adapté.

Dans le cas où le serveur d'annuaire ne trouve aucun service d'adaptation ou aucune chaîne d'adaptation, il renvoie le message "RIEN" au serveur A. Quand le serveur A reçoit ce message, il envoie le message "RIEN" au client, et le client donne une notification à l'utilisateur.

4.2. Tests

Dans cette partie, je donne les résultats d'implémentation de mon application. Dans ce test, j'ai utilisé l'image de ressource : « test.jpg ». J'utilise deux téléphone Androïd comme deux serveurs multimédias appelé le serveur A et le serveur B. Le serveur d'annuaire est un ordinateur. Le fichier XML qui présente la description des services sur le serveur multimédia B est présenté dans la table ci-dessous :

```
<?xml version="1.0" encoding="UTF-8"?>
<services>
    <service weight="2" actionType="transcodage" mediaType="image"
             className="ConvertImageJPG2PNG.java" out=".png" in=".jpg"
             name="ConvertImageJPG2PNG" id="1"/>
    <service weight="2" actionType="transcodage" mediaType="image"
             className="ConvertImagePNG2JPG.java" out=".jpg" in=".png"
             name="ConvertImagePNG2JPG" id="2"/>
    <service weight="2" actionType="transcodage" mediaType="image"
             className="ConvertImageJPG2GIF.java" out=".gif" in=".jpg"
             name="ConvertImageJPG2GIF" id="5"/>
    <service weight="2" actionType="transcodage" mediaType="image"
             className="ConvertImageGIF2TIF.java" out=".tif" in=".gif"
             name="ConvertImageGIF2TIF" id="6"/>
    <service weight="2" actionType="transmodage" mediaType="image2video"
             className="ConvertImageTIF2MOV.java" out=".mov" in=".tif"
             name="ConvertImageTIF2MOV" id="7"/>
    <service weight="2" actionType="transcodage" mediaType="video"
             className="ConvertImageMOV2AVI.java" out=".avi" in=".mov"
             name="ConvertImageMOV2AVI" id="8"/>
    <service weight="2" actionType="transmodage" mediaType="video2image"
             className="ConvertImageAVI2PNG.java" out=".png" in=".avi"
             name="ConvertImageAVI2PNG" id="9"/>
    <service weight="2" actionType="transmodage" mediaType="image2video"
             className="ConvertImageTIF2WMV.java" out=".wmv" in=".tif"
             name="ConvertImageTIF2WMV" id="10"/>
    <service weight="2" actionType="transmodage" mediaType="video2sound"
             className="ConvertImageWMV2MP3.java" out=".mp3" in=".wmv"
             name="ConvertImageWMV2MP3" id="11"/>
```

```
    <service weight="3" actionType="transmodage" mediaType="text2video"
                className="T2S.java" out=".wav" in=".txt" name="T2S" id="12"/>
</services>
```

Cadre 4.1 : Description des services sur le serveur B

La description des services sur le serveur A :

```
<?xml version="1.0" encoding="UTF-8"?>
<services>
     <service weight="1" actionType="transformation" mediaType="image"
                className="Resize.java" out=".jpg" in=".jpg" name="Resize" id="1"/>
      <service weight="1" actionType="transformation" mediaType="image"
                className="Resize.java" out=".png" in=".png" name="Resize" id="2"/>
</services
```

Cadre 4.2 : Description des services sur le serveur A

L'utilisateur demande une image avec des contraintes qui sont décrit dans le fichier « profil.xml ». Il envoie ce fichier au serveur A qui stocke l'image de ressource : « test.jpg » . Les serveurs A , B, le serveur d'annuaire vont chercher et exécuter les services pour satisfaire les contraintes de l'utilisateur. J'ai exécuté cinq cas de testes. Le premier cas est le plus simple. Le dispositif de l'utilisateur accepte les caractéristiques de la ressource. Donc on ne doit pas chercher et exécuter les services d'adaptation. Le deuxième cas est plus complexe. Le serveur d'annuaire doit chercher les services d'adaptation. Le serveur B exécute les services d'adaptation. Donc il faut échanger des données entre le serveur A et le serveur B. Dans le troisième cas on ajoute plusieurs valeur d'un attribut dans profil. Dans le quatrième cas, on ajoute des attributs différents au profil. Le dernière cas, le serveur d'annuaire ne trouve aucune chaîne de services d'adaptation.

1/ Cas 1 : L'utilisateur accepte des images JPG. Dans ce cas, il n'y a pas de problème parce que le format de l'image de ressource est JPG. Donc le serveur multimédia A envoie l'image de ressource à l'utilisateur. Les caractéristiques du profil sont présentées dans la figure 4.4 :

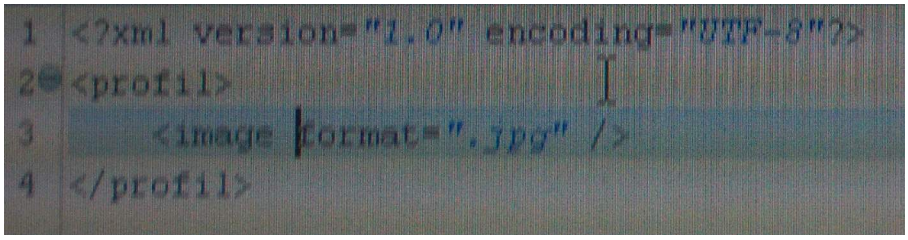

Figure 4.4 : Le contenu de profil dans premier cas de test

Le message sur le serveur A est dans la figure 4.5 :

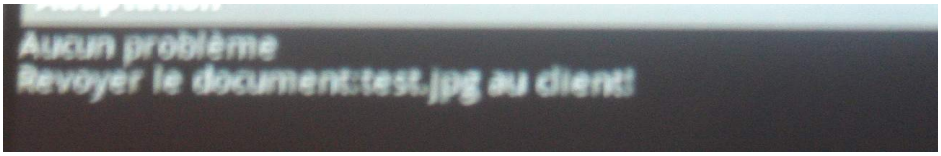

Figure 4.5 : Le message de serveur A dans premier cas de test

2/ Cas 2 : Dans ce cas, l'utilisateur accepte des images à la résolution 200x200. Donc, dans le fichier « profil.xml », on décrit l'attribut « resolution » et sa valeur 200x200 comme la figure 4.6 :

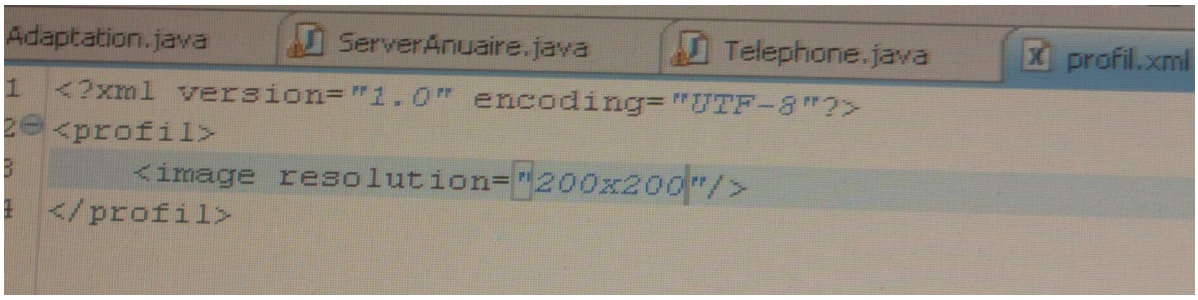

Figure 4.6 : Le contenu de profil dans deuxième cas de test

L'utilisateur envoie ce fichier au serveur multimédia A. Comme il ne peut pas changer la résolution d'image, il envoie le problème de changement de la résolution au serveur d'annuaire. Le serveur d'annuaire trouve les résultats comme dans la figure 4.7 :

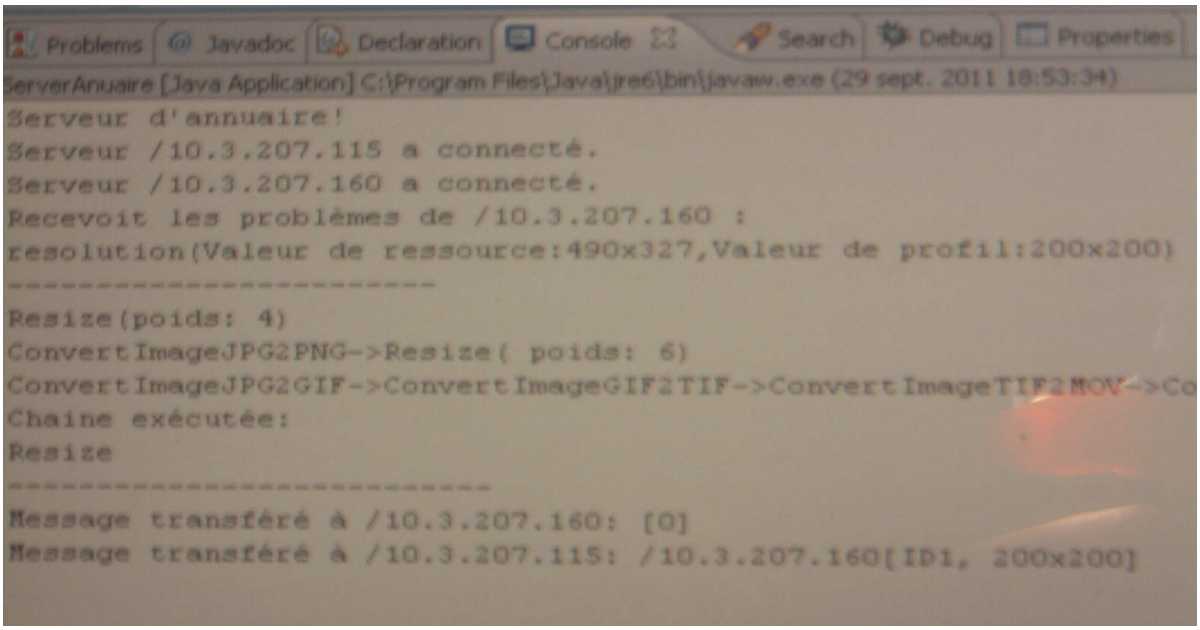

Figure 4.7 : Les résultats sur le serveur d'annuaire dans deuxième cas de teste

En se basant sur les poids, le serveur d'annuaire choisit la chaîne qui consiste seulement en un seul service : « Resize » qui est exécuté sur le serveur B. Donc le serveur d'annuaire envoie au serveur A le message « [0] », et au B le message : « /10.3.207.160[ID1,

200x200] » (l'adresse IP de A , l'ID du service Resize et la valeur du profil).

Après avoir reçu le message du serveur d'annuaire, le serveur B reçoit la ressource qui est stockée dans le serveur A pour créer une nouvel image après application du service Resize. Et puis, le serveur B renvoie cette nouvelle image au serveur A. Les messages sur le serveur B et le serveur A sont illustrés dans les figures 4.8 et 4.9 :

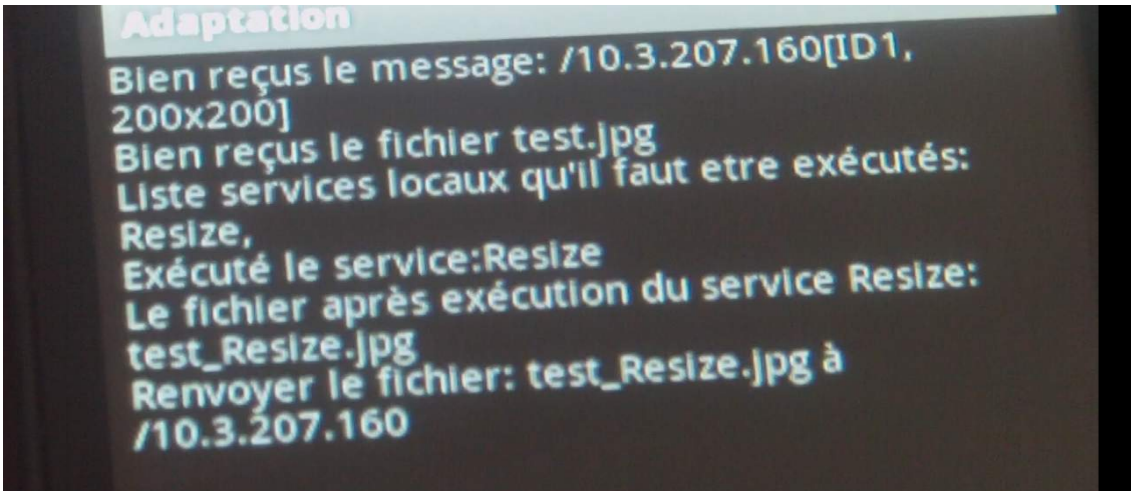

Figure 4.8 : Le message de serveur B dans deuxième cas de teste

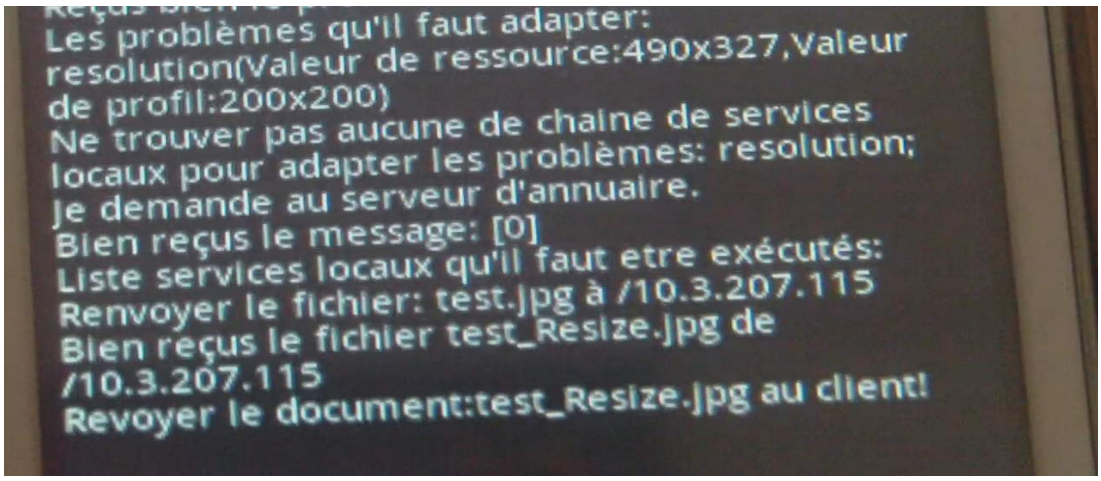

Figure 4.9 : Le message de serveur A dans deuxième cas de teste

3/ Cas 3 : Dans ce cas, on test le cas de multiple valeurs de profil. L'utilisateur veut l'image à résolution de 200x200 ou de 100x100. Le contenu du profil est décrit dans la figure 4.10 :

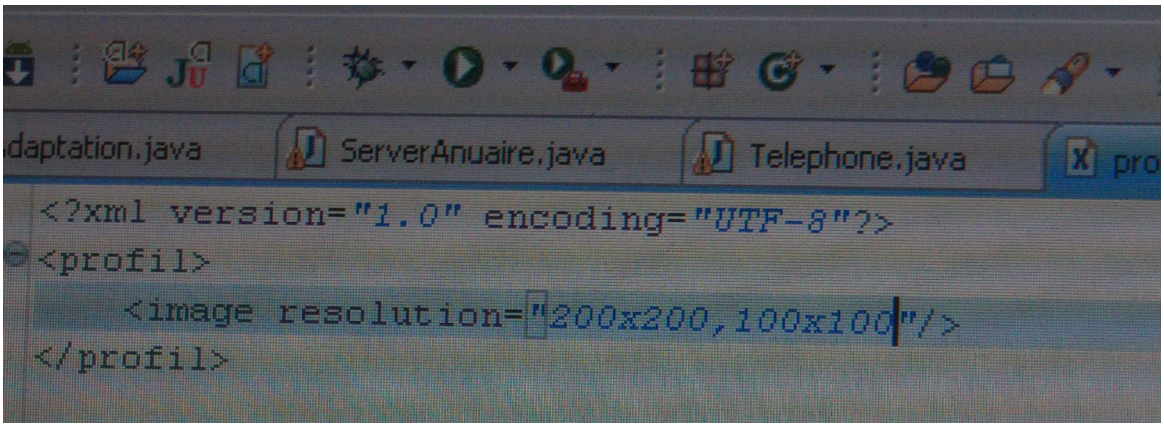

Figure 4.10 : Le contenu de profil dans troisième cas de teste

Comme le premier cas, le serveur d'annuaire choisit la meilleure chaîne qui consiste un service : « Resize » comme dans la figure 4.11 :

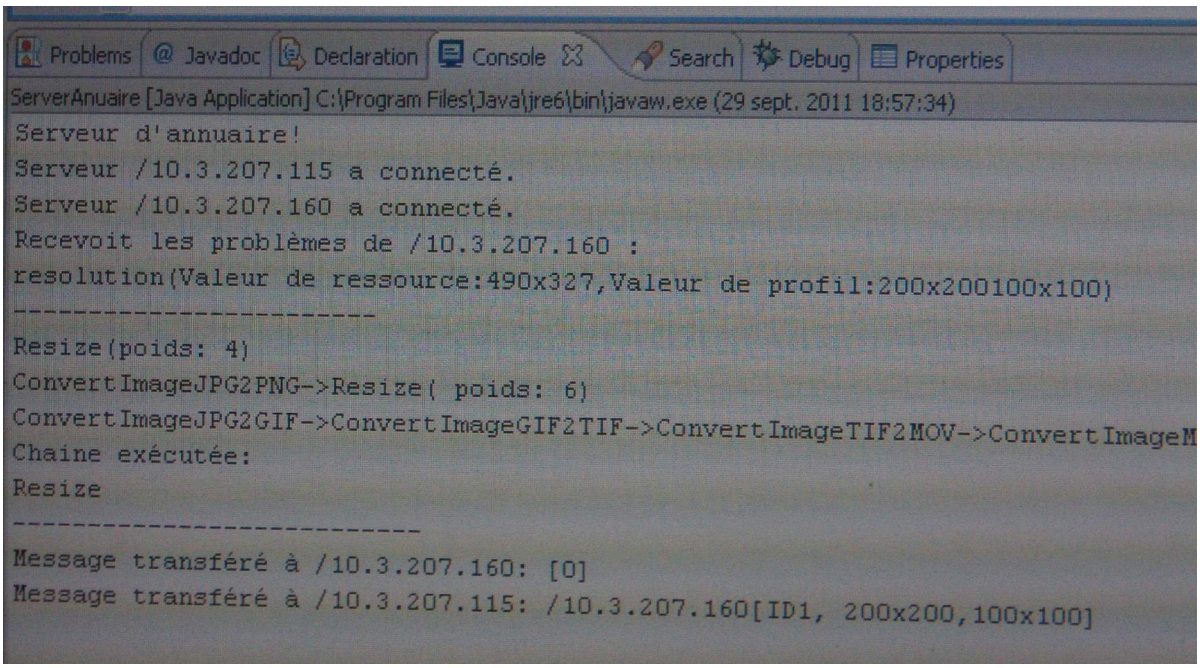

Figure 4.11 : Les résultats sur le serveur d'annuaire dans troisième cas de teste

Les messages sur le serveur B et le serveur A sont montrés dans les figures 4.12 et 4.13 :

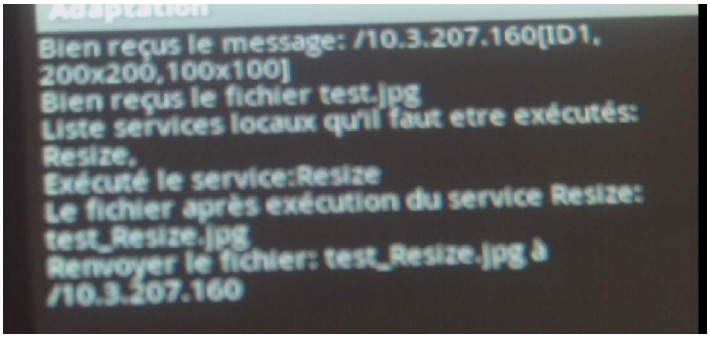

Figure 4.12 : Le message de serveur B dans troisième cas de teste

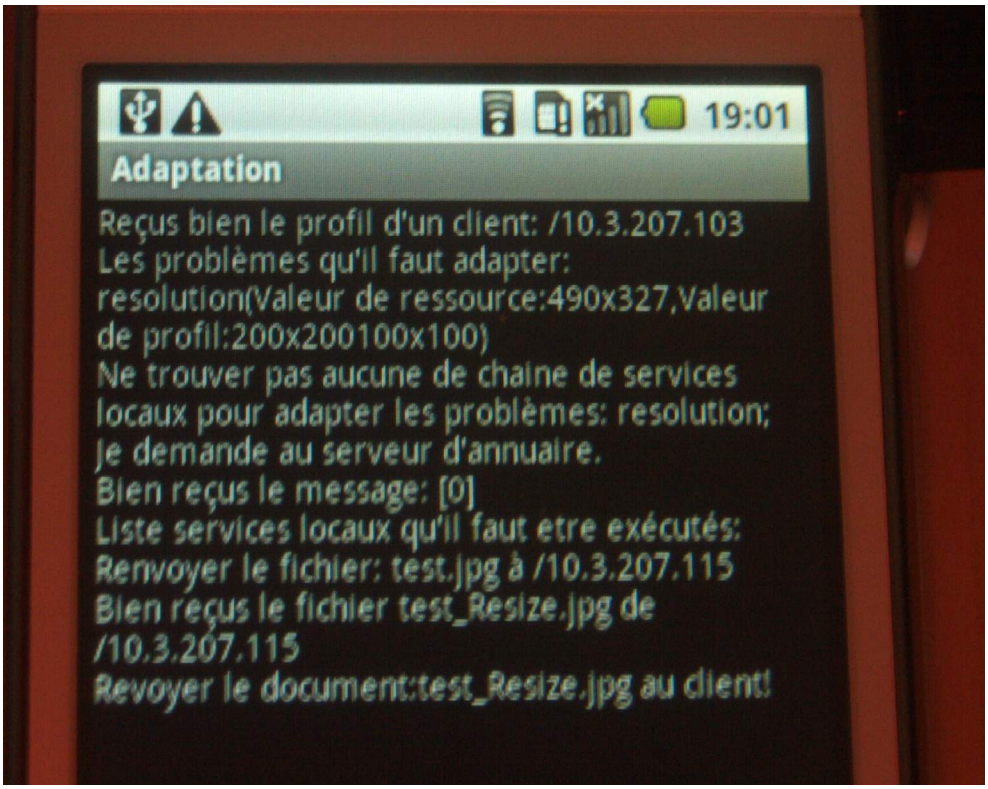

Figure 4.13 : Le message de serveur A dans troisième cas de teste

4/ Cas 4 : Dans ce cas, on test les multiple problèmes d'adaptation. L'utilisateur veut des images au format PNG et à la résolution 200x200. Le contenu du profil est présenté dans la figure 4.14 :

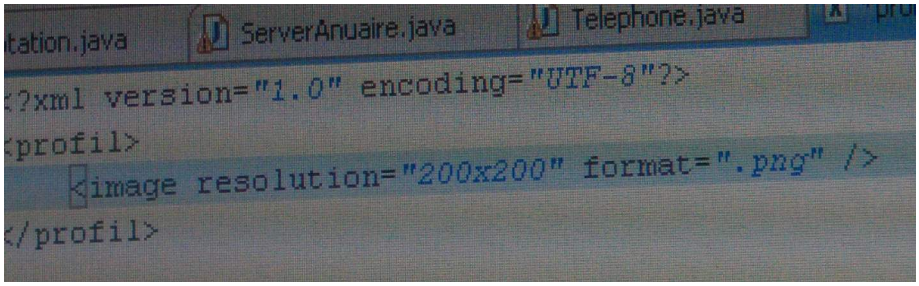

Figure 4.14: Le contenu de profil dans le quatrième cas de test

Dans ce cas, il faut résoudre deux problèmes : le problème de changement de format qui est adapté par le service ConvertImageJPG2PNG sur le serveur A, et le problème de changement de résolution qui est adapté par le service Resize sur le serveur B. Donc le serveur d'annuaire trouve les chaînes comme dans la figure 4.15:

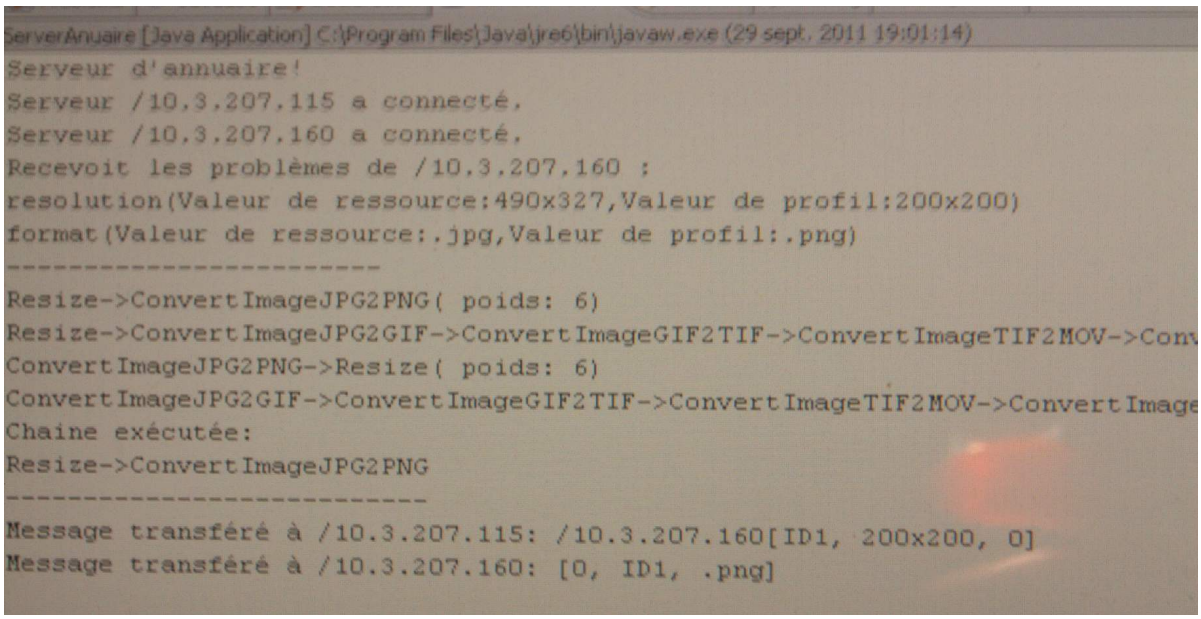

Figure 4.15 : Les résultats sur le serveur d'annuaire dans quatrième cas de test

Les messages sur le serveur B :

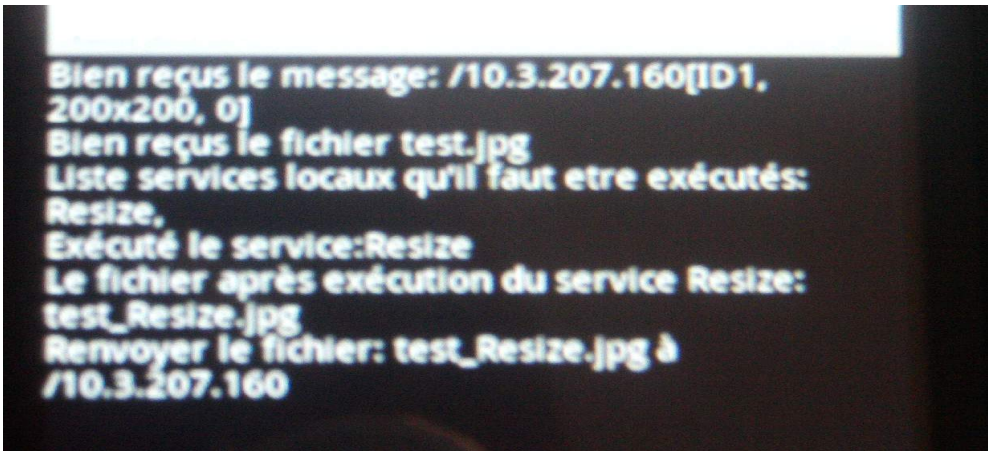

Figure 4.16 : Le message de serveur B dans quatrième cas de test

Les messages sur le serveur A :

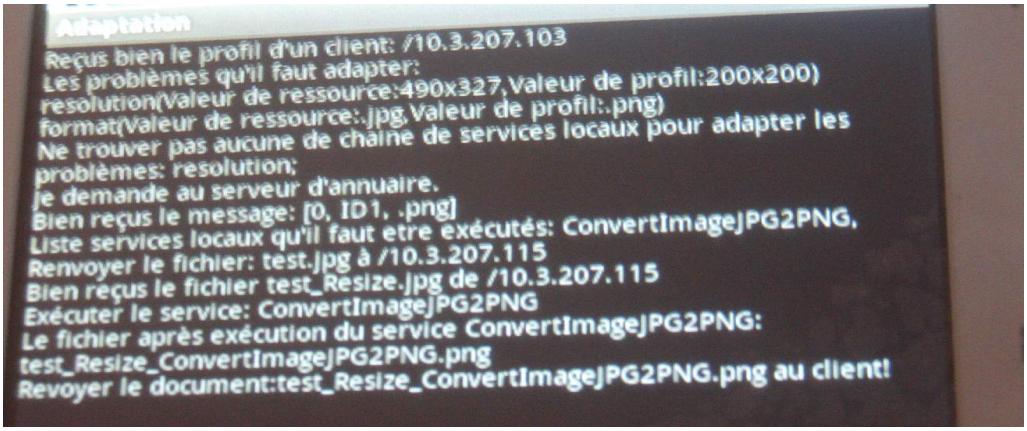

Figure 4.17: Le message de serveur A dans quatrième cas de teste

5/ Cas 5: On essaye le cas où on ne peut pas adapter toutes les contraintes. L'utilisateur veut

une image à la résolution 200x200, au format PNG et au niveau de gris : 100. Le contenu du profil est montré dans la figure 4.18 :

```
<?xml version="1.0" encoding="UTF-8"?>
<profil>
    <image resolution="200x200" format=".png" size="100" color="100"/>
</profil>
```

Figure 4.18: Le contenu de profil dans le cinquième cas de test

Comme il n'y a pas de service qui adapte le problème de couleur sur les deux serveurs multimédias A et B, le serveur d'annuaire ne trouve aucune chaîne d'adaptation comme la figure 4.19 :

```
Serveur d'annuaire!
Serveur /10.3.207.115 a connecté.
Serveur /10.3.207.160 a connecté.
Recevoit les problèmes de /10.3.207.160 :
resolution(Valeur de ressource:490x327,Valeur de profil:200x200)
format(Valeur de ressource:.jpg,Valeur de profil:.png)
size(Valeur de ressource:41482,Valeur de profil:100)
color(Valeur de ressource:255,Valeur de profil:100)
-----------------------
Ne trouve pas aucune chaine de services!!!
```

Figure 4.19 : Les résultats sur le serveur d'annuaire dans le cinquième cas de test

Le serveur d'annuaire renvoie au serveur A un message pour notifier qu'il ne trouve aucune chaîne des services d'adaptation. Et puis, le serveur A affiche un message de notification. Le message de serveur A est présenté dans la figure 4.20 :

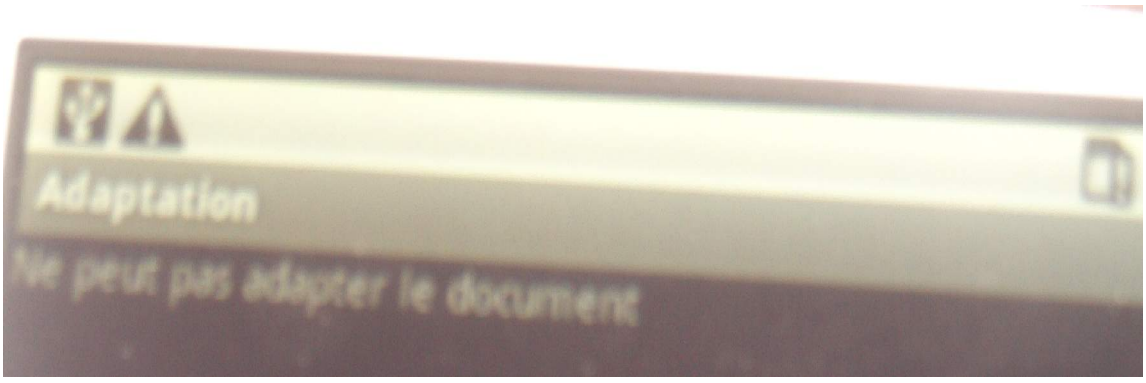

Figure 4.20: Le message de serveur A dans le cinquième cas de test

Après cinq scénarios de test, le système d'adaptation marche bien. Les résultats sont exacts. On peut ajouter facilement des contraintes au profil et des valeurs de caractéristiques. Le système détecte bien des problèmes et cherche bien des services d'adaptation. Donc, on peut dire que le système satisfait le but de ce stage. Dans la partie suivante, je représente

l'évaluation des performances du système.

4.3. Évaluation des performances

Dans les algorithmes de recherche (avant et après), quand on visite un noeud, on cherche le noeud suivant jusqu'à ce que l'on trouve un noeud satisfaisant les règles de recherche (on l'appelle noeud final) ou un noeud qui n'a aucun noeud suivant.

Définition : le chemin de recherche est une chemin du nœud initial au nœud qui n'a aucun nœud suivant ou au noeud qui satisfait les règles de recherche.

Supposons que le nombre de nœuds suivants de chaque nœud dans l'arbre de recherche est la distribution uniforme.

On appelle :

- Le nombre de nœud suivant est N.
- Le nombre de nœud dans une chemin est L.
- Le nombre de nœud dans une chaîne des services est R.

Le temps pour visiter tous les nœuds dans l'arbre de recherche est N*L. Quand on trouve un nœud final dans une chaîne de services, on ne s'intéresse pas à ses nœuds suivants, alors le temps moyen pour chercher des services est : N*L – (L – R)*N = R*N

On trouve qu'avec un ensemble de services, le temps pour chercher des services dépend le nombre de services dans une chaîne.

Maintenant, je présente les résultats expérimentaux de performance de l'algorithme de recherche. Dans ce test, l'algorithme de recherche est exécuté sur le Smartphone LG540 qui utilise le système Androïd 2.1. Le temps de recherche est varié à cause du chargement du système d'exploitation, alors je répète le test pour obtenir le résultat moyen.

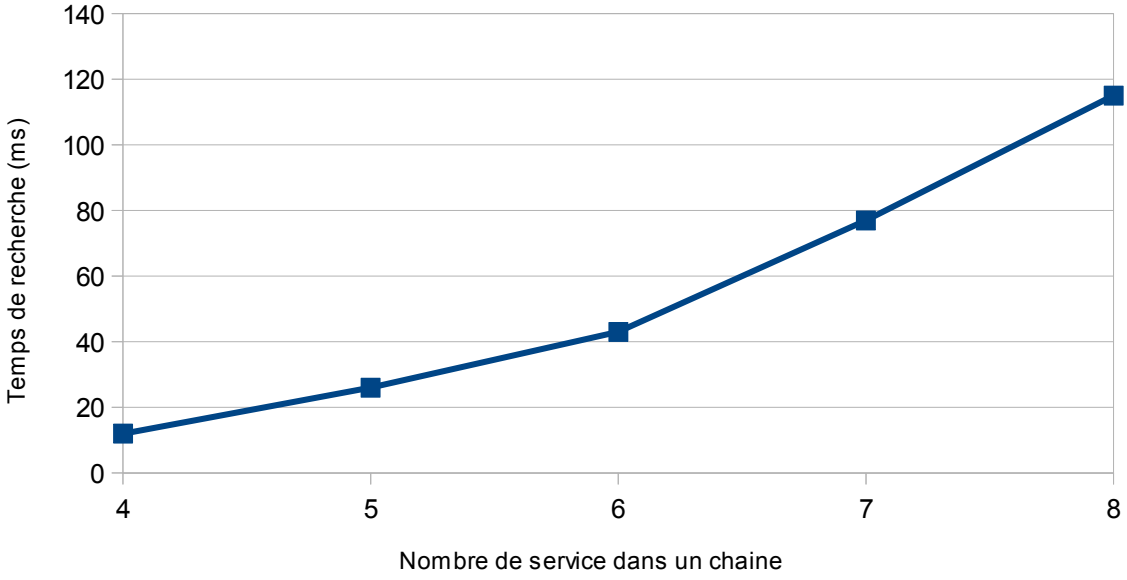

Figure 4.21: Le temps de chercher des chaînes de services

En consultant la figure 4.21, le temps de recherche augmente linéairement. Dans l'algorithme de recherche, on cherche des chaînes de services qui résolvent le problème de changement de format tout d'abord. Ensuite, on ajoute d'autres services qui résolvent d'autres problèmes aux chaînes trouvées. La figure ci-dessous présente l'évaluation quand on dispose de quelques problèmes.

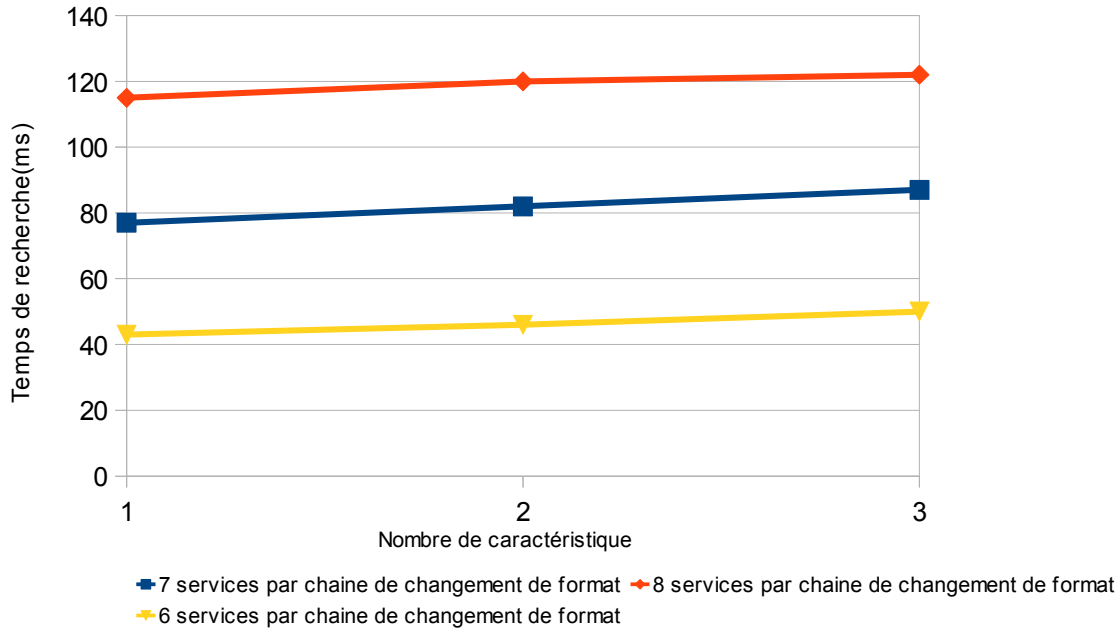

Figure 4.22: Le temps de recherche des chaînes qui résolvent tous les problèmes

Dans cette évaluation, j'essaye d'ajouter des services aux chaînes qui changent le format de ressources. En regardant la figure, quand on ajoute des caractéristiques dans le profil, le temps à rechercher des chaînes de services qui résolvent tous les problèmes augmente peu. En effet le temps à trouver une position convenable dans une chaîne pour ajouter le service est beaucoup moins important que le temps à chercher une chaîne dans l'arbre de recherche.

Chapitre 5: Conclusion

5.1. Conclusion

Ce stage vise à mettre en place une architecture d'adaptation de document multimédia. Précisément, les travaux demandés dans ce stage sont: (1) proposer un algorithme de recherche des services d'adaptation ; (2) de proposer en conséquence une stratégie de déploiement des processus d'adaptation (exprimant où le document ou bien une partie du document doit être adapté) ; (3) évaluer le temps d'adaptation.

Dans mon stage, j'ai proposé une architecture d'adaptation de document multimédia et le stratégie d'adaptation. J'ai aussi créé une application d'adaptation qui exécute l'algorithme de recherche des services d'adaptation. J'ai testé mon application dans cinq scénarios ou mon application marche bien. De lus, j'ai fait une évaluation pour obtenir les résultats expérimentaux de performance de l'algorithme de recherche. Le résultat obtenu est une architecture permettant de satisfaire toutes les demandes de l'utilisateur.. Comme les demandes des utilisateurs sont souvent variées, et le type de documents multimédias augmente de plus en plus, alors j'ai construit une architecture extensible et la flexible.

Extensibilité : l'extensibilité est présentée sous deux respects. Premièrement, l'architecture peut élargir le nombre de services d'adaptation. Chaque serveur multimédia stocke un fichier de format XML qui décrit des descriptions des services étant exécutés sur le serveur multimédia. Donc il est facile d'élargir la collection de services. On peut ajouter un nouveau service au système en décrivant une nouvelle description du services au fichier de description des services. On peut également supprimer un service dans le système en supprimant la description du service. Donc des services d'adaptation peuvent être fabriqué par un third-party. Et puis, ils sont ajoutés facilement au système. Deuxièmement, on peut appliquer l'architecture pour tous les types multimédias. Le système extrait facilement des informations de documents multimédias pour de la vidéo, du son, de l'image et du texte.

Flexibilité : l'utilisateur décrit ses demandes dans le fichier profil.xml. Donc, c'est facile de le modifier. L'utilisateur peut ajouter des nouvelles caractéristiques, modifier des caractéristiques, ou supprimer quelques caractéristiques dans le fichier facilement et rapidement. Donc l'utilisateur peut demander le document multimédia correspondant à n'importe quel contexte et à n'importe quel dispositif cible.

Comme l'architecture dépend des services d'adaptation pour déterminer des types d'adaptation, quand l'utilisateur spécifie une contrainte pour laquelle il n'existe pas de service d'adaptation pour la résoudre, on n'adapte pas la ressource, c'est-à-dire l'utilisateur

n'obtient pas de document multimédia adapté. Par exemple : l'utilisateur veut un texte en anglais, mais on a seulement un texte en français sur le serveur multimédia, et il n'existe pas de service de traduction dans tous les serveurs multimédias, l'utilisateur ne va rien obtenir.

5.2. Perspectives

Dans le future, nous améliorons notre approche en basant sur les standards du W3C. Nous devrons proposer une architecture logicielle ouverte à base de services (SOA) adaptée à une exécution de type cloud computing. Ce type d'adaptations étant énergivores et les dispositifs mobiles impliqués limités en énergie et puissance de calcul, une prise en compte énergétique dans les adaptations devra également être mise en œuvre. Ces améliorations seront déployées dans notre thèses.

Annexe

L'algorithme de recherche des services d'adaptation :

```
Algorithme : chercherService( Pbrs, SD)
Entrée : la liste de problèmes d'adaptation Pbrs, la liste de services disponibles SD,
Sortie :  liste de chaîne L
// les variables
   isAdded    //
   isExecuted
   isFoundBefore , isFoundAfter   //
   S          // service
   Pbr        // problème
   ListeTemp   //Liste de chaîne
   ListeChaineAvant  //Liste de chaîne de résultat dans la recherche par avant
   ListeChaineApres  //Liste de chaîne de résultat dans la recherche par après
//  initialisation
     L = Ø
     ListeTemp = Ø
    isAdded = false
    isExecuted = false
    isFoundBefore = false
    isFoundAfter = false
    isFound = false
 //  commencement
   for( Pbr ∈ Pbrs ) {
       // si pbr change le format de ressource
       if( Pbr ∈ PCF ) {
            for( S ∈ SD ) {
                 if( (S ∈ SCF) ∧ (O.S ∈ PV.Prb) ) ) {
                      if(I.S = = V.Prb) {
                                C = C ∪ {S}              // ajouter S à une chaîne
                                L = L ∪ {C}              // ajouter la chaîne à liste
                           EXE.Prb = true        //pbr est exécuté
```

```
                    }
                    else {
                        if( chercherAvant(S, PV.Prb, L )) EXE.Prb = true   //pbr est exécuté
                    }
                }
                //Si Prb n'est pas exécuté
                if( !EXE.Prb) break ;
            }
        }
// traiter les services qui ne changent pas le format de ressource
// S'il existe au moine une chaîne qui résout le problème de change le format
    if (  L ≠ ϕ  ) {
        for(  Pbr ∈ Pbrs  ) {
            // si Pbr ne change pas le format de ressource
            if (  Prb ∉ PCF  ) {
                for(  S ∈ SD  ) {
                    // Si S satisfait touts les caractéristiques de Prb
                    if(  (O.S ∈ PV.Prb) ∧ (A.S = A.Prb) ∧ (M.S = M.Prb) ∧ (N.S = N.Prb)  )
                    {
                        // ajouter s à chaque chaîne dans liste de chaîne
                            for(  C ∈ L  ) isAdded = ajourterService( S, C )
                            if (  isAdded ) EXE.Prb = true    //pbr est exécuté
                    }
                }
                //Si Prb n'est pas exécuté
                    if( !EXE.Prb ) break ;
            }
        }
    }
// S'il n'y a pas des problèmes changeant le format
    else {
        // On cherche seulement les chaînes de service qui adaptent le première problème
        // Les services qui résolvent les autres  problèmes sont ajoutés à ces chaînes
```

```
Prb = Prbs [0] ;
for( S∈SD ) {
    // Si S satisfait touts les caractéristiques de Prb
   if( (O.S∈FP.Prb)∧(A.S=A.Prb)∧(M.S=M.Prb)∧(N.S=N.Prb) )
    {
        isFoundBefore = chercherAvant(S, FR.Prb, ListeChaineAvant)
        if( (I.S≠FR.Prb)∧isFoundBefore∧(FP.Prb≠Ø) )
        {
            isFoundAfter = chercherApres(S, FP.Prb, ListeChaineApres)
            if( isFoundBefore && isFoundAfter )
            {
                //On a deux types de chaîne de service: chaîne avant et chaîne
                // après. Il faut mélanger ces chaînes pour obtenir les chaînes
                // complètes
                for( C∈ListeChaineAvant )
                    L = melangerChaine(C, ListeChaineApres)
                EXE.Prb = true ;
            }
        }
    }
}
// Traiter les autres problèmes
// On ajoute les services qui résolvent les problèmes aux chaînes dans L
for( Prb∈Prbs )
{
    if( Prb≠Prbs[0] )
    {
        for( S∈SD )
        {
            if( (O.S∈PV.Prb)∧(A.S=A.Prb)∧(M.S=M.Prb)∧(N.S=N.Prb) )
            {
                for( C∈L )
                {
```

```
                    isAdded = ajourterService( S, C )
                    if(isAdded)
                    {
                        EXE.Prb = true
                        isAdded = false
                    }else {
                        Sz est la dernière service dans C
                        isFoundBefore =
                            chercherAvant(S, O.Sz , ListeChaineAvant)
                        if( (FP.Prb ≠ Ø) ∧ (O.Sz ≠ FR.Prb) ∧ isFoundBefore )
                        {
                            isFoundAfter =
                                chercherApres(S, FP.Prb, ListeChaineApres)
                            if( isFoundAfter)
                            {
                                for( C1 ∈ ListeChaineAvant )
                                    Liste = melangerChaine(C1, ListeChaineApres)
                                    L= melangerChaine(C, Liste)
                            }
                            }else if( (FP.Prb = Ø) ∧ isFoundBefore ){
                                L= melangerChaine(C, ListeChaineAvant)
                                EXE.Prb = true
                                isFoundBefore = false
                        }
                    }
                }
            }
        }
    // Si on ne cherche pas de service ou les services trouvés ne sont pas
    // ajoutables aux chaînes dans L
    if( !EXE.Prb) break
    else{
     // Si on ajoute Prb à quelques chaînes dans L, il faut supprimer les
```

```
                    // chaînes qui ne contiennent pas Prb
                        for( C∈L )
                        {
                            if( Prb∉C ) L=L\{C}
                        }
                    }
                }
            }
        }
        Retourner L
}
```

Fonction d'addition d'un service à une chaîne de services

```
ajouterService( S, C )
Entrée : service S, la chaîne de services C
Sortie : true si on ajoute S à C avec succès
          false si inversement
// variables
   isAdded
// initialisation
   isAdded = false
// commencement
// vérifier chaque service dans la chaîne C
for( c∈C ) {
    if(O.S = I.c) {
        ajouter S avant c dans C
        isAdded = true
        break
    }
    // Si ne pas encore ajouter S à la chaîne C
    if(I.S = O.c){
```

```
            ajouter S après c dans C
            isAdded = true
            break
        }
    }
```

Fonction de mélange de deux chaînes

melangerChaine(C,L1)

Entrée : une liste de chaîne de services : L1

Sortie : une liste de chaîne L

// initialisation

$L = \emptyset$

// commencement

for($C1 \in L1$)

{

$C = C \cup C2$

$L = L \cup \{C\}$

}

Retourner L

Algorithme de recherche par avant

Algorithme : chercherAvant(S, F ,L)

Entrées : service initial S, l'ensemble de format F, liste de chaîne L

Sortie : **true** si L n'est pas vide, **false** si L est vide

// les variables

Stack ListeService //utilisé pour enregistrer des traces, c'est-a-dire on enregistre

// chaque nœud visité dans listeService

isFound //la valeur de retourner

$Parent \in S$

// initialisation

isFound = false

$ListeService = \emptyset$

// commencement

```
if( I.S ∉ F ) ListeService ∪ S //laisser s au sommet de ListeService
while( ListeService ≠ Ø ){
  prendre le nœud de sommet de ListeService : NœudVisite
  EXE.NœudVisite = true //le nœud NœudVisite est invité
  Parent = chercherParent(NœudVisite)
  if( Parent = ϕ ) supprimer le NœudVisite dans le ListeService
  else{
      EXE .Parent = true
      NœudVisite = NoeudCheck( Parent ) //le nœud duquel on part au nœud Parent est
                                        // NœudVisite
      Laisser Parent au sommet de ListeService
      if( I.Parent ∈ F ) {
          Copier les éléments de ListeService en chaîne de services C
          Inverser les éléments dans C // ranger les éléments dans C
          L = L ∪ {C} // ajouter la chaîne à liste de chaîne
          Supprimer le parent dans le ListeService
          isFound = true // trouver une chaîne
      }
  }
  Retourner isFound
}
```

```
Algorithme : chercherParent(Sa)
Entrées : service initial Sa
Sortie : service final Sz
//les variables
  S // un service
// commencement
  for( S ∈ Avant(Sa) ){
    if( (EXE.S = false) ∨ (NoeudCheck(S) ≠ Sa) ) {
        Sz = S
        break
```

```
    }
  Retourner  Sz
 }
```

L'algorithme de recherche par après

```
Algorithme : chercherApres(S, F, L)
Entrées : service initial S, l'ensemble de format  F, liste de chaîne L
Sortie : true si L n'est pas vide, false si L est vide
// les variables
   Stack ListeService   //utilisé pour enregistrer des traces, c'est-a-dire on enregistre
                        //chaque  nœud visité dans ListeService
   isFound   //la valeur de retourner
    Enfant∈S
// initialisation
  isFoud = false
  ListeService =   ϕ
// commencement
if(  O.S∉F  ) laisser s au sommet de ListeService
while(  ListeService≠Ø  ){
   prendre le nœud de sommet de ListeService : NœudVisite
   EXE.NœudVisite  = true  //le nœud  NœudVisite est invité
   Enfant = chercherEnfant(NœudVisite)
   if( enfant =  ϕ  ) supprimer le NœudVisite dans la liste ListeService
   else{
       EXE.Enfant = true
       NoeudCheck( Enfant) = NœudVisite    //le nœud duquel on part au nœud enfant est
                                           //  nœudVisite
       Laisser  Enfant au sommet de ListeService
       if(   O.Enfant∈F  ) {
            Copier les éléments de ListeService en chaîne de services  C
             L=L∪{C}      // ajouter la chaîne à liste de chaîne
            Supprimer le enfant dans  ListeService
             isFound = true
```

```
        }
    }
    Retourner isFound
}
```

Algorithme : chercherEnfant(S_a)
Entrées : service initial S_a

Sortie : service final S_z

```
//les variables
    s∈S       // un service
// commencement
    for(   s∈Apres(Sa)   ){
        if(  (EXE.s= false)∨(NoeudCheck(s)≠Sa)  ) {
            Sz = s
            break
        }
     Retourner  Sz
  }
```

Références

[1] Les formats du document multimédia: http://developer.android.com/guide/appendix/media-formats.html

[2] Frank Ableson, Charlie Collins, Robi Sen: Unlocking Android, *Manning Publications Co*, 2009

[3] Florent Garin, Sylvain Wallez: Développer des applications mobiles pour les Google Phones, *Dounod, Paris*, 2009

[4] Reto Meier: Professional Android Application Developpement, *Wiley Publishing, Inc.*,2009

[5] Parcour d'arbre, disponible sur: http://zanotti.univ-tln.fr/ALGORITHMIQUE/PARCOURS-GRAPHE.html

[6] La thèse de Sébastien Laborie: Adaptation sémantique de documents multimédia, *l'Université Joseph Fourier - Grenoble 1*, 28 Mai 2008

[7] Zhijun Lei and Nicolas D. Georganas: Context-based Media Adaptation in Pervasive Computing , *In Proceeding of Canadian Conference on Electrical and Computer Engineering*, Mai 2001

[8] Marriott, K., Meyer, B., and Tardif, L: *Fast and efficient client-side adaptivity for SVG*. *In Proceedings of WWW,* 2002

[9] Harini Bharadvaj, Anupam Joshi and Sansanee Auephanwiriyakul: An ActiveTransco ding Proxy to Supp ort Mobile Web Access, *17th IEEE Symposium on Reliable Distributed Systems,* 1998

[10] Margaritis Margaritidis, George C. Polyzos :Adaptation techniques for ubiquitous Internet multimedia, *Wireless Communications and Mobile Computing,* 2001