

---

# Kalinahia : modèle de qualité de service pour les applications multimédia reconfigurables

Sophie Laplace—Marc Dalmau—Philippe Roose

LIUPPA - IUT Informatique de Bayonne  
Château Neuf – Place Paul Bert  
64100 Bayonne - France  
{laplace, dalmau, roose}@iutbayonne.univ-pau.fr

---

*RÉSUMÉ. L'un des défis actuels des Systèmes d'Information est d'assurer la transmission de données semi structurées telles les données multimédia dans un environnement distribué et ubiquitaire c'est-à-dire pervasif. Ils doivent alors garantir aux utilisateurs une qualité de service assurant aussi bien l'accessibilité aux données quelles que soient les conditions matérielles et la disponibilité du réseau que la cohérence de ces informations et en particulier leur intelligibilité ce qui impose une personnalisation du service. Dans ce cadre, nous proposons une méthode de conception basée sur des modèles originaux d'applications multimédia et de qualité de service. Nous définissons également Kalinahia une plate-forme de supervision utilisant une heuristique centrée sur l'utilisateur qui permet de définir à tout instant quelle configuration de composants logiciels répond le mieux aux vœux de l'utilisateur en terme de service.*

*ABSTRACT. One of the current challenges of Information Systems is to ensure semi structured data transmission such as multimedia data in a distributed and ubiquitous i.e. pervasive environment. They must then guarantee to the users a quality of service ensuring data accessibility whatever the material and networks conditions are. They also must guarantee information coherence and particularly intelligibility that imposes a personalization of the service. Within this framework, we propose a design method based on original models of multimedia applications and quality of service. We also define Kalinahia a supervision platform using a user centred heuristics who allows to define at any moment which configuration of software components constitutes the best answers to user's wishes in term of service*

*MOTS-CLÉS : qualité de service, conception, multimédia, applications réparties, adaptation, reconfiguration, composants.*

*KEYWORDS: quality of service, design, multimedia, distributed application, adaptation, reconfiguration, components.*

---

## 1. Introduction

L'un des défis actuels des Systèmes d'Information est de rendre disponibles de plus en plus d'informations multimédia et de leur associer de plus en plus de services. En outre, les utilisateurs exigent de ces services qu'ils soient accessibles sur des terminaux toujours plus mobiles, ubiquitaires et miniaturisés. Cependant, fournir un service n'est pas suffisant, car la qualité du service est essentielle pour obtenir un succès commercial, tout particulièrement dans le multimédia. En effet, le grand public se désintéressera inexorablement des applications multimédia mobiles si elles ne conservent pas un niveau satisfaisant de qualité de service — QoS — lors de leur utilisation.

Or la mise à disposition d'un grand volume de données semi structurées ayant des contraintes temporelles sur un réseau aussi imprédictible et hétérogène qu'Internet ne peut reposer sur les mécanismes classiques de réservation de ressources. Il est alors nécessaire de proposer une adaptation dynamique de ces systèmes d'informations et des traitements associés. Dans ce cadre, nous avons étudié une démarche de conception basée sur des modèles originaux d'applications multimédia et de QoS. Nous avons également défini une plate-forme de supervision utilisant une heuristique centrée sur l'utilisateur permettant de choisir à tout instant quelle configuration de composants logiciels répond le mieux aux vœux des utilisateurs en terme de service.

Lors de cette étude nous avons considéré que la QoS n'incluait pas seulement les aspects de transport et de traitement de l'information mais aussi ceux concernant son intelligibilité. En effet notre approche permet de mettre en œuvre des restructurations de l'application aussi bien lors de modifications du contexte matériel que du contexte humain comme, par exemple, lorsque la langue utilisée n'est pas compréhensible par certains utilisateurs.

Dans cet article, nous situons tout d'abord notre problématique par rapport à quelques travaux significatifs. Puis nous présentons nos modèles de QoS et d'applications multimédia réparties. Enfin nous proposons le modèle KaliNahia de plate-forme d'exécution — KalitateaNahia soit rechercher la qualité en basque — permettant d'optimiser la QoS et présentons les résultats les plus significatifs obtenus par notre simulateur.

## 2. Etat de l'art

La QoS a été introduite initialement dans le domaine des réseaux pour décrire la qualité du service fourni aux applications par les systèmes de communications. Puis, son acception s'est étendue (CCITT, 1989) (Vogel *et al*, 1995). Elle a quitté l'aspect purement technique pour rejoindre des préoccupations proches de l'utilisateur.

Or cette qualité dépend du contexte d'exécution de l'application (Rakotonirainy *et al*, 2001) (Stephen *et al*, 2002) (Ayed *et al*, 2004), en particulier lorsque celui-ci est variable et à plus forte raison s'il varie de façon imprévisible. Il existe alors deux moyens d'assurer une certaine QdS à l'utilisateur : adapter le contexte à l'application, ou du moins contraindre le contexte, et adapter l'application au contexte, y compris à l'utilisateur. Or il n'est pas toujours possible d'agir sur le contexte surtout lorsque l'on y inclut, comme nous le faisons, les utilisateurs eux-mêmes. Ainsi, si dès sa conception et sa réalisation, l'adaptation de l'application au contexte a été prise en compte, l'utilisateur pourra bénéficier de la meilleure QdS possible.

Pour nos travaux, nous avons choisi comme domaine d'étude celui qui nous semble le plus représentatif de cette problématique : les applications multimédia réparties sur Internet. En effet, elles sont caractérisées (Hafid *et al*, 1998) à la fois par de grandes exigences de qualité, par une grande sensibilité au contexte et par la forte variabilité de ce dernier. Elles utilisent un grand volume de données soumis à des contraintes temporelles aussi bien de délai que de synchronisation — vidéosurveillance, médecine augmentée, vidéoconférence. Les solutions habituellement proposées pour assurer une QdS suffisante aux systèmes d'informations multimédia dans un contexte variable utilisent soit l'allocation de ressources soit une forme d'adaptation dynamique au contexte qui peut concerner le réseau, l'intergiciel, les flux multimédia, l'application, les composants de l'application ou plusieurs de ces éléments à la fois (Gu *et al*, 2002).

D'autre part, les intergiciels constituent un outil efficace pour maintenir une certaine QdS. Ainsi CALiF Multimédia (Garcia, 2001) propose un intergiciel pour applications coopératives multimédia en utilisant l'allocation de ressources du réseau et l'adaptation des besoins de l'application aux ressources disponibles. L'intergiciel Agilos (Li *et al*, 2001) permet un contrôle hiérarchique de la QdS en utilisant non seulement la réservation de ressources mais aussi la configuration de composants. JQoS (Zhu *et al*, 2001) propose une application de vidéoconférence sur Internet avec adaptation des flux multimédia selon l'état des performances de QdS du système. Enfin, QuO (Schantz *et al*, 2003) (Wang *et al*, 2004) permet la spécification et la gestion de la QdS dans des applications construites à base de composants répartis : la plate-forme et l'application sont adaptées.

Désireux de placer l'utilisateur au centre des préoccupations de QdS, nous proposons un modèle d'intergiciel permettant d'adapter dynamiquement les applications multimédia réparties à leur contexte d'exécution afin de maintenir une QdS optimale pour les utilisateurs, cette qualité étant définie individuellement : le modèle Kalinahia. Cet intergiciel, lui-même réparti, ajoute ou supprime des composants et reconfigure ou restructure les assemblages de composants formant l'application.

### 3. Modélisation de la QdS et des applications multimédia réparties

Nous proposons une gestion de la QdS adaptée aux systèmes d'informations multimédia sur Internet. C'est pourquoi nous présentons un modèle de QdS centré sur l'utilisateur. A partir de ce modèle, nous avons ensuite construit un modèle d'application multimédia répartie.

#### 3.1. Modélisation de la QdS des applications multimédia réparties

Nous proposons un modèle original de la QdS des applications multimédia sur Internet et nous définissons celle-ci de la manière suivante :

**Définition 1 :** *La qualité de service est l'adéquation entre le service souhaité par l'utilisateur et le service qui lui est fourni.*

Elle peut être modélisée de façon plus simple que dans le cas général des applications réparties (Firesmith, 2003) car, dans une application multimédia, seules comptent les caractéristiques que l'utilisateur peut directement percevoir. Ainsi nous nous limitons à deux niveaux hiérarchiques (Gu *et al*, 2002), les caractéristiques et les critères :

- les caractéristiques décrivent des paramètres simples de la QdS mesurables, tels la cadence vidéo ou le temps de transmission, ou non mesurables, comme la langue utilisée dans un discours.
- les critères de QdS regroupent les caractéristiques en fonction de leur dépendance — critère contextuel Co — ou non — critère intrinsèque In — vis-à-vis du contexte.

Ainsi nous regrouperons dans le critère intrinsèque des caractéristiques telles que la taille d'un écran, la taille des images d'un flux ou la langue alors que nous regrouperons dans le critère contextuel des caractéristiques telles que le temps de transmission ou la cadence vidéo. De cette manière, notre modèle souligne l'importance du contexte et de ses variations imprédictibles dans les applications multimédia sur Internet en ne se restreignant pas aux seules caractéristiques des réseaux.

Pour représenter la QdS d'une entité, nous utilisons une note pour chacun des deux critères. Ces critères étant orthogonaux, un choix de valeurs des qualités extrêmes — 0 pour une qualité rédhibitoire pour l'utilisateur et 1 pour une qualité optimale — permet de représenter la QdS par le point du plan ayant ces deux valeurs pour coordonnées dans un repère orthonormé (Figure 1). Ainsi, nous conservons toute la richesse de la représentation du modèle qui discerne l'influence du contexte.

Nous avons ensuite exploité l'analogie entre cette QdS et la notion d'utilité en micro-économie (Parkin, 1992) (DaSilva, 2000). La similitude entre les courbes d'indifférence qui définissent le degré de satisfaction des consommateurs en fonction

des biens consommés et les équipotentielles de QdS, nous a conduits à choisir comme modèle d'évaluation de la QdS la fonction donnant à cette qualité la valeur du pire critère. Ce choix nous permettra d'établir une comparaison entre deux configurations afin de rechercher celle offrant la QdS optimale. Ces configurations seront définies à l'aide du modèle d'applications multimédia réparties que nous présentons dans le paragraphe à venir.

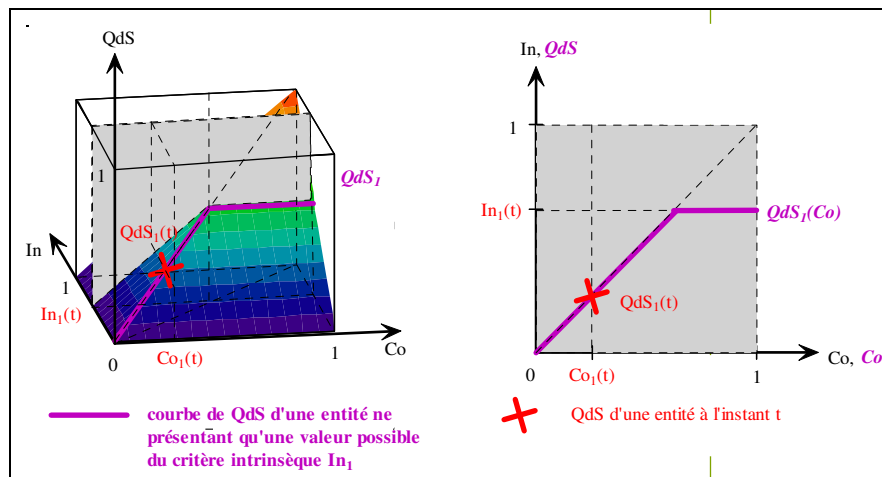


Figure 1. Représentation de la QdS :  $QdS = \min(In, Co)$

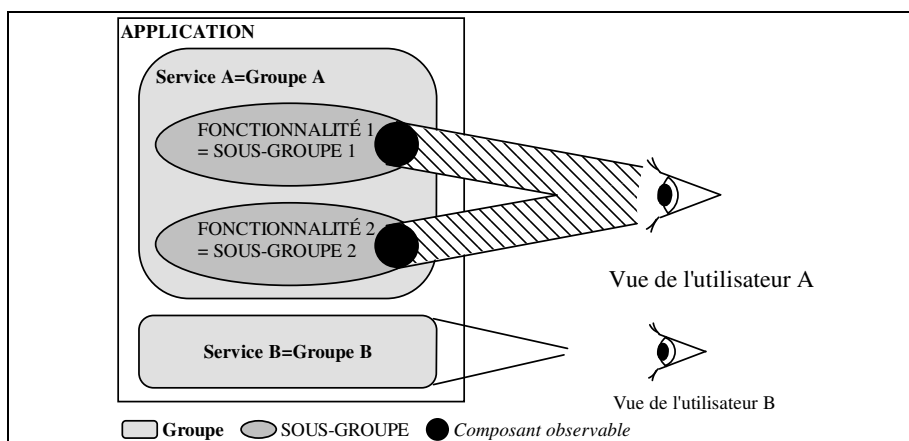
### 3.2. Modélisation des applications multimédia réparties

En raison des contraintes de développement imposées aux applications destinées au grand public, nous proposons d'exécuter les applications multimédia réparties sur une plate-forme. Nous définissons par conséquent un modèle d'application venant s'associer à une plate-forme d'exécution. Du point de vue structurel, l'application est construite en fonction de la vision qu'a l'utilisateur du service qui lui est fourni — le Groupe. Chaque service est composé de différentes fonctionnalités — les Sous-Groupes (Figure 2). Ces derniers sont constitués de composants, logiciels, matériels ou humains, reliés par des flux d'information. Les composants logiciels sont encapsulés dans des Processeurs Élémentaires tandis que les flux de données sont transportés par des Conduits (Dalmau *et al*, 2005). Les Conduits et les Processeurs Élémentaires permettent de conserver la synchronisation des flux quels que soient les conditions de leur transport et les traitements qu'ils subissent.

Ainsi dans une vidéoconférence, au moins deux Groupes cohabiteront : l'un permettant aux intervenants de discuter et l'autre aux spectateurs d'assister à cette

conférence. Le Groupe des téléspectateurs contiendra généralement deux Sous-Groupes leur permettant de percevoir les informations sonores d'une part et visuelles d'autre part. Selon les composants utilisés la QdS des Sous-Groupes sera modulable : le discours pourra être transmis sous forme sonore ou à l'aide de sous-titre. Le son sera alors synchronisé avec l'image car ces deux informations seront transmises au travers des mêmes Conduits.

En effet, nous faisons le choix de poser la conservation de la synchronisation des flux comme une condition sine qua non à leur transmission. Cette optique se justifie par les caractéristiques des applications multimédia où les différents flux, en particulier le son et l'image, s'unissent pour contenir la sémantique des informations.

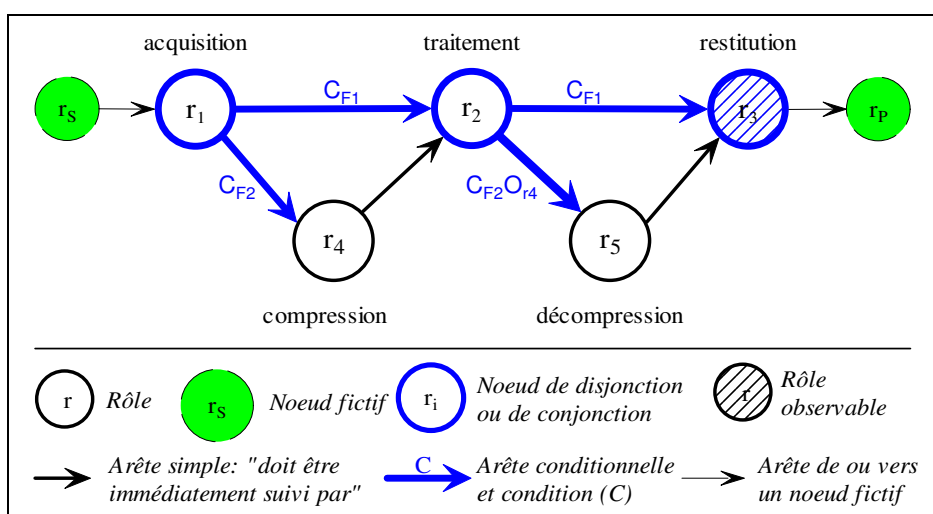


**Figure 2.** *Modèle structurel d'une application multimédia répartie*

Grâce à ce modèle, l'analyse de l'application se fait en termes de services rendus à l'utilisateur i.e. les Groupes. Cette vision de haut niveau permet de faire abstraction des contraintes matérielles et logicielles pour se concentrer sur la réalisation d'une application multimédia viable et utilisable. D'autre part, le concept de Sous-Groupe permet de traiter l'adaptation à l'environnement en faisant abstraction des contraintes de localisation puisque les composants le constituant sont répartis sur les différents hôtes de l'application. Ce modèle issu d'une analyse descendante permet donc de traiter en amont la gestion de la QdS puis de s'intéresser aux fonctionnalités de bas niveau de l'application avant de procéder à l'implantation des composants. Nous pouvons donc parler d'**une conception centrée QdS**.

### 3.3. Représentation des applications multimédia et de leur QdS

Nous proposons une représentation unifiée des applications et de leur QdS sous forme de graphes orientés polaires (Eles, 2000). De façon générale, les nœuds de ces graphes sont associés aux éléments opérationnels de l'application — rôles — alors que les arêtes réifient les flux de données (Figure 3). Ces graphes permettent d'identifier les assemblages de composants et de flux communs à plusieurs configurations et d'optimiser ainsi l'évaluation en factorisant celle de ces parties communes.



**Figure 3.** Graphe des flots de contrôle d'un sous-groupe transmettant des images

Les arêtes conditionnelles permettent principalement d'indiquer des informations de deux sortes:

- des possibilités de choix amenant à l'utilisation d'une configuration ou d'une autre de l'application : choix d'une décomposition fonctionnelle — assemblage ordonné de rôles —, choix d'un composant, choix d'une localisation. Ainsi, les conditions  $C_{F1}$  et  $C_{F2}$  du graphe de la Figure 3 expriment la possibilité de choisir d'appliquer à un flux d'images soit seulement un traitement —  $C_{F1}$  — soit un traitement et une compression/décompression —  $C_{F2}$ .

- des contraintes liées à un choix précédent : un rôle en impose un autre, un composant en impose un autre comme l'utilisation d'une compression sur un flux d'images impose l'utilisation d'une décompression ce qui est précisé par la condition  $O_{r4}$  dans l'exemple de la Figure 3.

Les graphes que nous proposons ont deux fonctions principales. La première est d'aider le concepteur dans sa tâche en lui permettant de représenter l'application aux différents stades de sa réalisation. La seconde est de proposer un outil de représentation et d'évaluation des applications à la plate-forme lui permettant ainsi de les superviser.

Leur construction suit donc les étapes de la conception de l'application. Le premier graphe, le graphe des flots de contrôle, (Figure 3) définit l'ordre d'exécution des traitements appliqués aux flux d'informations. Puis les éléments le composant sont graduellement affinés de manière à aboutir à un graphe fonctionnel en une analyse fonctionnelle descendante. Sont alors définis les rôles atomiques — rôle réalisable par un seul composant de l'application — et les échanges de données de l'application.

A partir des informations fournies par le concepteur — composants disponibles, contraintes de localisation — le graphe fonctionnel est complété de manière à décrire, pour chaque rôle atomique, les différents choix de composants et de localisation possibles ainsi que la présence de Conduits et de Processeurs Élémentaires. C'est le graphe de configuration qui est alors obtenu.

Comme chaque nœud de ce graphe représente un élément qui influe sur la QdS, il est alors possible d'en dériver le graphe d'évaluation qui détermine l'algorithme de définition de la QdS de l'application en remplaçant chaque composant par son influence sur la QdS. Ce graphe permet en particulier de déterminer pour chaque Sous-Groupe les caractéristiques de sa QdS : taille des images, langue utilisée, temps de traitement, cadence vidéo... Chaque caractéristique est ensuite comparée avec les vœux de l'utilisateur de manière à lui attribuer une note. Celui-ci aura attribué aux caractéristiques un poids relatif qui permet ainsi de définir la QdS du Sous-Groupe par une moyenne pondérée des notes des caractéristiques. Les notes de QdS de l'application et des Groupes sont obtenues à l'aide de moyennes pondérées similaires. Notre choix s'est porté sur des moyennes arithmétiques pondérées car ce sont les plus usuelles et les plus faciles à pondérer par l'utilisateur mais nous sommes conscients qu'il serait intéressant d'affiner cette étude et d'envisager d'autres outils mathématiques (moyennes géométrique, harmonique, quadratique etc.).

Notre évaluation de la QdS repose donc sur la récolte des préférences de l'utilisateur. Or il n'est pas possible de lui demander la totalité des informations qui permettraient de différencier et classer toutes les configurations car cela représenterait une quantité d'informations trop importante. En particulier, plus les configurations à ordonner diffèrent par une entité de bas niveau architectural plus le nombre de configurations à étudier est grand. Nous proposons donc que le concepteur définisse, en fonction de l'application, le niveau le plus bas sur lequel la plate-forme proposera à l'utilisateur d'exprimer ses préférences.

L'utilisateur précisera ses vœux de service de différentes façons en fonction de la complexité du niveau concerné. Ainsi il pourra ordonner toutes les possibilités pour

des complexités faibles comme les Groupes. Il paramètrera le système d'évaluation pour des complexités un peu plus élevées et la plate-forme se chargera de générer les notes et d'ordonner les possibilités étudiées. En revanche, pour les complexités trop élevées, de type exponentiel, il ne lui sera pas possible d'exposer ses préférences.

Notre système de représentation permet de passer de manière isomorphe des spécifications fonctionnelles à l'implantation de l'application puis à l'évaluation de sa QdS. Elle permet donc d'automatiser certaines tâches du développement. Par son choix, nous souhaitons ainsi préserver la possibilité de proposer une plate-forme aidant le concepteur dans sa tâche même si ce n'est pas l'objet de nos travaux actuels. En effet, ceux-ci portent sur la plate-forme d'exécution permettant de gérer la QdS pendant le fonctionnement de l'application.

#### **4. Modélisation de la plate-forme Kalinahia**

Nous présentons un modèle de plate-forme d'exécution appelé Kalinahia qui prend en charge de manière répartie à la fois le déploiement de l'application et sa supervision (Figure 4). Nous proposons que la plate-forme reconfigure dynamiquement l'application en adaptant sa composition et son implantation aux variations du contexte d'exécution de manière à optimiser sa QdS. Pour ce faire, chacun des constituants de la partie opérative de l'application — composants et flux — produit des événements de reconfiguration à l'intention de la plate-forme dès qu'il détecte une variation de son contexte d'exécution.

Lorsqu'une reconfiguration est nécessaire, la plate-forme doit proposer une configuration offrant une meilleure QdS. Il s'agit donc, a priori, de trouver un assemblage optimum de composants. Or ce problème est connu pour être NP-complet dans le cas général (Garey *et al*, 1979)( Kuipers *et al*, 2002).



notes du critère contextuel soient elles aussi identiques. Or si les critères intrinsèque et contextuel ont les mêmes notes, les deux configurations ont la même note de QdS. Nous pouvons donc dire que deux configurations fournissent un service proche si et seulement si leurs notes des critères intrinsèque et contextuel sont proches ce qui implique que les notes de QdS soient proches. Nous remarquerons que ceci n'implique en rien que la configuration la plus proche au sens du service soit celle dont la note de QdS est la plus proche.

Finalement, la plate-forme commencera donc sa recherche par l'évaluation des configurations ayant la même note de critère intrinsèque que la configuration actuelle. Elle se contentera donc dans un premier temps de modifier la note du critère contextuel.

Si la nouvelle configuration n'est pas optimale, la plate-forme en sera informée par le biais de nouveaux événements de reconfiguration ce qui permettra d'améliorer la QdS par une nouvelle recherche et d'atteindre ainsi, de proche en proche, la meilleure configuration. La Figure 5 illustre ce principe mis en œuvre par la plate-forme pour la recherche itérative d'optimum. Il est apparenté à la méthode du recuit simulé utilisée en particulier dans des problèmes de placement et de routage en électronique.

La recherche d'une configuration meilleure se fait en étudiant successivement des ensembles finis de configurations ayant des services proches ; les familles. Chaque famille fournit un service de même nature et possède la même note de critère intrinsèque : elles ne se différencient donc que par leur adaptabilité au contexte. La plate-forme recherche tout d'abord des solutions parmi les configurations de même critère intrinsèque avant d'évaluer la QdS des configurations ayant un critère intrinsèque différent comme dans l'exemple de la Figure 6.

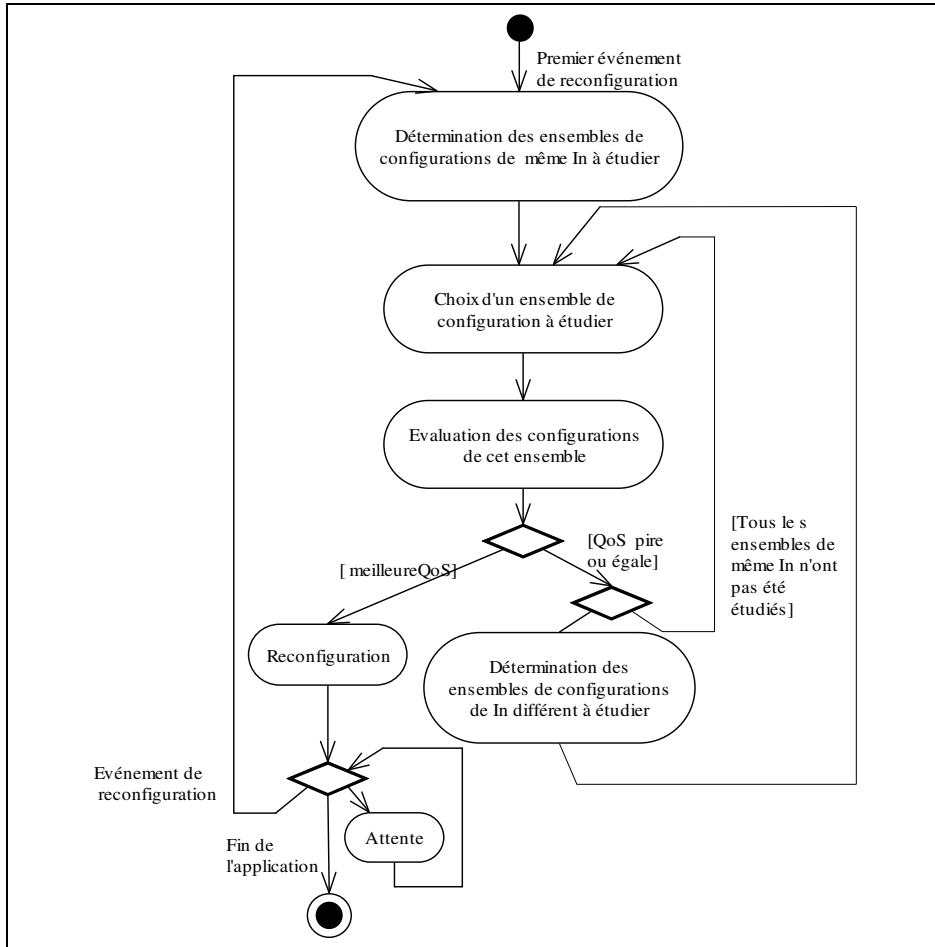


Figure 5. Principe des reconfigurations par la plate-forme Kalinhia

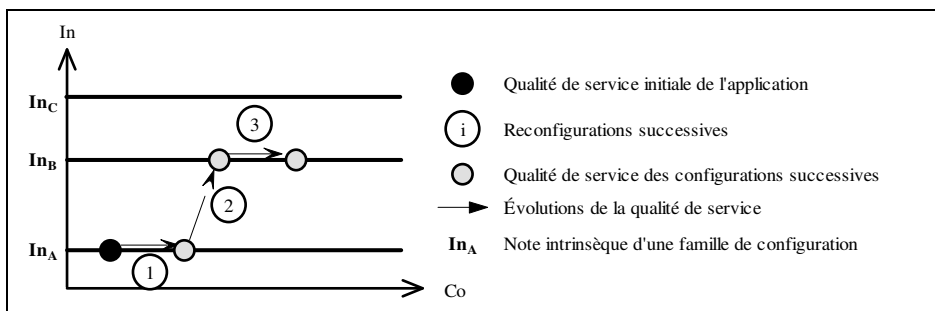


Figure 6. Exemple d'évolution de la QoS lors des reconfigurations

Pour être efficace, la plate-forme doit cibler les modifications à réaliser sur l'application. Dans la mesure où chaque constituant de la partie opérative de l'application est susceptible de générer des événements de reconfiguration, l'information obtenue par la plate-forme est très précise et lui permet de savoir quels composants de l'application sont problématiques et quelles entités doivent être modifiées ou supprimées. Ainsi, dans un premier temps la plate-forme va pouvoir restreindre l'ensemble d'étude aux configurations qui ne diffèrent de celle en cours d'exécution que par le composant à l'origine de l'événement de reconfiguration. Cependant lorsque cette approche n'apporte aucune solution, se pose le problème du déploiement ex-nihilo d'un Sous-groupe ou d'un Groupe.

Les événements de reconfiguration peuvent être issus des Conduits et des Processeurs Élémentaires qui effectuent des mesures passives ou actives sur les composants et les flux de manière à détecter les baisses ou les améliorations de QdS possibles.

Ils peuvent également être émis par des composants appelés espions que le concepteur de l'application a introduits pour récolter des informations contextuelles non mesurables comme la langue d'un locuteur lors d'une vidéoconférence (Laplace, 2006). Cette information pourra par exemple indiquer que le service n'est plus adapté pour un auditeur ne comprenant pas cette langue et imposera donc une reconfiguration qui devra proposer une traduction ou des sous-titres. Nous proposons que le concepteur associe un composant espion à chaque caractéristique intrinsèque de service qui lui semblera opportune. Ce composant compare l'état de la caractéristique avec les critères optimaux que peut fournir l'application en fonction des vœux de l'utilisateur et génère un événement de reconfiguration s'il y a une différence. L'événement produit induit des contraintes sur les solutions à trouver comme l'utilisation obligatoire de tel composant ou de telle fonctionnalité ou la suppression d'une entité. Il est à remarquer que des composants espions sont également utilisés pour détecter l'arrivée ou le départ d'un utilisateur et participent de ce fait au déploiement de l'application en provoquant la création ou la suppression d'une instance de Groupe.

#### ***4.2. Modèle de la plate-forme d'exécution Kalinahia***

Nous proposons un modèle de plate-forme, Kalinahia, mettant en œuvre une heuristique itérative qui améliore la QdS à chaque itération. La recherche d'une configuration meilleure repose alors sur deux critères.

Le premier est imposé par les contraintes temporelles des applications multimédia: c'est l'efficacité de la plate-forme. Il est nécessaire que la plate-forme réagisse rapidement afin d'éviter les ruptures du service. Ceci est obtenu par la génération d'événements de reconfiguration.

Le second critère vient, lui aussi, des particularités des applications multimédia où la perception que l'utilisateur a de l'application est centrale pour évaluer la QdS : c'est le maintien de la continuité ergonomique lors d'une reconfiguration appelée plasticité (Coutaz, 2001). Il est respecté grâce à l'étude de la proximité de service entre configurations. Celle-ci est déterminée, d'une part, en utilisant l'architecture que nous avons conçue de manière à ce qu'elle reflète la vision qu'a l'utilisateur du service et, d'autre part, en utilisant les vœux que l'utilisateur exprimera.

Toutefois la plate-forme rencontre aussi des situations particulières comme lors du déploiement ex-nihilo d'un Groupe ou d'un Sous-Groupe. Dans ces cas-là, nous proposons trois solutions. La première consiste à prendre en compte les configurations par défaut définies par le concepteur lorsqu'elles sont disponibles. La deuxième repose sur l'évaluation par la plate-forme de toutes les possibilités lorsque cela est réalisable. Enfin, la troisième consiste en l'implantation de la configuration la moins coûteuse en ressources systèmes ou de la configuration la plus souvent utilisée.

A partir de tous ces critères, nous avons construit une heuristique (Figures 7 et 8). Nous avons défini sa complexité en temps de calcul et déterminé qu'elle était polynomiale. Elle ne dépend plus que de la complexité intrinsèque de l'application et de l'étendue de l'offre de QdS que le concepteur souhaite proposer. Selon toute logique, cette complexité là est maintenant incompressible.

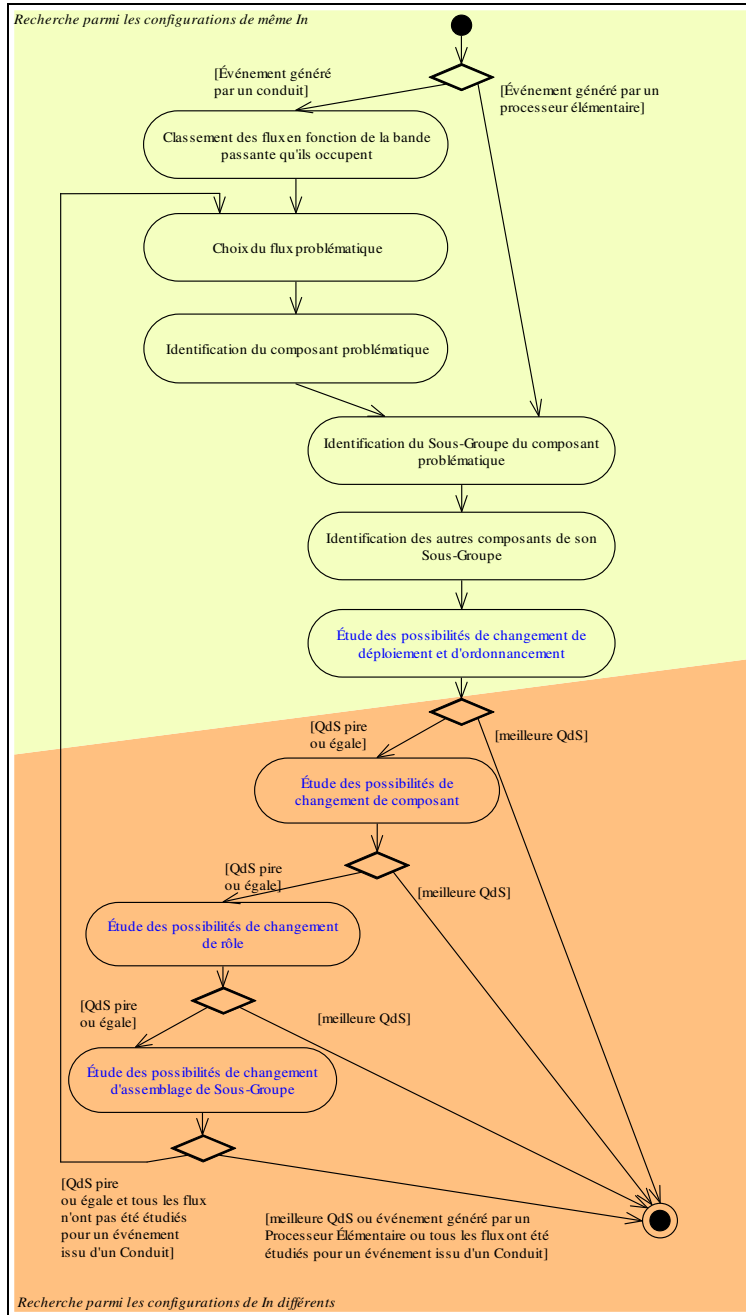


Figure 7. Recherche provoquée par un composant de l'application

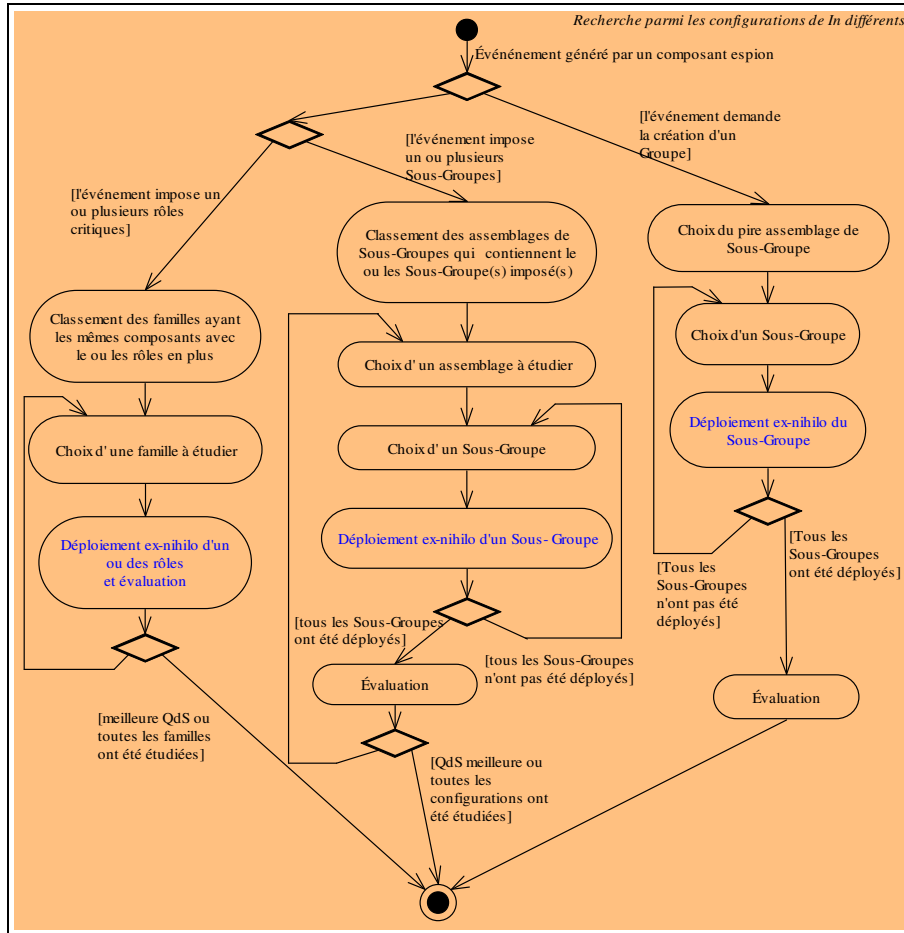


Figure 8. Recherche provoquée par un composant espion

## 5. Implantation et simulation de la plate-forme Kalinahia

### 5.1. Implantation de la plate-forme

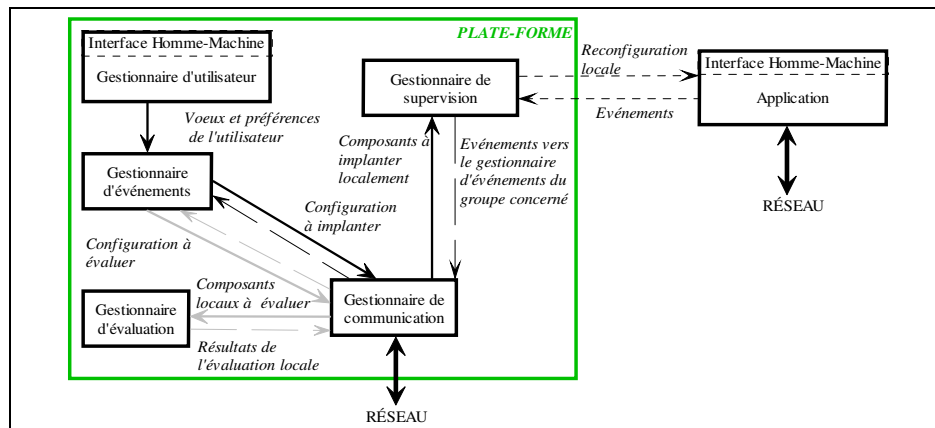
L'efficacité de l'heuristique que nous avons proposée pour définir les reconfigurations dépend en partie du choix de l'événement qui permet d'identifier le composant problématique. C'est pourquoi nous avons élaboré un modèle de gestionnaire d'événements qui permet à la plate-forme d'intervenir sur la partie de

l'application la plus critique pour l'utilisateur. Nous avons ensuite déterminé que le traitement en parallèle des événements pouvait être réalisé soit au niveau d'un Sous-Groupe soit au niveau d'un Groupe en fonction des composants communs car le parallélisme est limité, d'une part, par notre volonté d'assurer la plasticité de l'application et, d'autre part, par l'impossibilité d'implanter un même composant de deux manières différentes en dehors de toute duplication.

Puis nous avons proposé un modèle structurel de la plate-forme (Figure 9) permettant la gestion répartie des événements mais également de l'évaluation dans le souci de respecter les contraintes temporelles des applications multimédia.

Sur tous les postes utilisés par l'application, la partie locale de la plate-forme est composée de cinq gestionnaires :

- le gestionnaire d'événements réalisant le choix de l'événement à traiter ;
- le gestionnaire d'évaluation pour évaluer les configurations ;
- le gestionnaire de communication assurant la communication entre les différentes parties de la plate-forme ;
- le gestionnaire d'utilisateur permettant la saisie des vœux de l'utilisateur ;
- le gestionnaire de supervision surveillant et reconfigurant l'application.



**Figure 9.** Structure de la plate-forme Kalinahia

## 5.2. Simulation de la plate-forme

Nous avons validé ce modèle grâce à un simulateur de la plate-forme développé avec Labview de National Instruments. Nous avons choisi ce dernier car c'est un

outil de simulation axé supervision et flux de données. Notre simulateur permet à la fois de simuler l'application et son contexte d'exécution.

L'application est construite à l'aide de composants qui ne réalisent aucune fonction sur les données multimédias. Leur exécution est simulée par l'évolution des caractéristiques de QdS des flux en sortie à partir des caractéristiques de QdS des flux reçus en entrée. Ces flux sont représentés par leurs caractéristiques de QdS. L'évaluation de la QdS est alors simulée par la définition des caractéristiques de QdS des flux fournis en sortie de l'application à un instant donné ainsi que par leur notation. La simulation de l'exécution d'une application est obtenue par l'estimation en continu des caractéristiques de QdS de l'ensemble des flux présents dans l'application (Figure 10).

Tableau des flux de l'application						
	Taille Im	Coul Im	Tps	Cad Vid	Débit	Coef Comp
FA			0,00E+0		0,00E+0	
FC			0,00E+0		0,00E+0	
FA1			1,00E+0		1,00E+0	1,00E+0
FC1	128*196	16 millions	1,00E+0	25	2,01E+1	1,00E+0
FC2a	128*196	16 millions	2,00E+0	25	2,01E+1	1,00E+0
FC3a	64*98	16 millions	2,20E+1	25	5,02E+0	1,00E+0
FA2a			1,00E+3		1,00E+0	1,00E+0
FC4a	64*98	256	1,05E+3	1,99E+1	1,00E+0	1,00E+0
FA3			1,00E+3		1,00E+0	1,00E+0
FC5	64*98	256	1,05E+3	1,99E+1	1,00E+0	1,00E+0
FA4			1,00E+3		1,00E+0	1,00E+0
FC6	64*98	256	1,05E+3	1,99E+1	1,00E+0	1,00E+0

**Figure 10.** Simulation de l'exécution d'une application

Notre logiciel permet de simuler l'état du réseau et des postes informatiques utilisés par l'application. En cours de simulation, il est ainsi possible de faire varier la bande passante disponible entre deux postes ainsi que le temps de transmission associé. La saturation d'un poste est simulée par des seuils représentant le débit maximal des flux locaux et le délai minimal de transmission en local. L'utilisateur du prototype peut également définir un coefficient utilisé pour multiplier le temps de traitement de référence des composants d'un poste et pour diviser le débit de référence de ces composants — ce coefficient est supérieur ou égal à un car cette valeur correspond à la fréquence d'horloge du poste.

L'appel dynamique des composants a pu être réalisé grâce à l'utilisation d'un modèle unique de composant pour programmer tous les composants de l'application. Le simulateur réifie également les Conduits et les Processeurs Élémentaires. Ainsi, il crée, supprime, déplace des composants, des Conduits, des Processeurs Élémentaires de la même façon que le ferait une plate-forme réelle avec des délais comparables à ce que donnerait une plate-forme comme OSGi (OSGi, 2007). Enfin, les graphes de l'application sont représentés par des matrices d'adjacence.

Pour fonctionner, le simulateur a besoin d'informations fournies par le concepteur de l'application et par les utilisateurs.

Le concepteur de l'application a préalablement défini :

- les Groupes, les Sous-Groupes, les assemblages possibles de Sous-Groupes ;
- les différentes décompositions fonctionnelles disponibles pour chaque Groupe et les familles qu'ils constituent ;
- les composants disponibles et les rôles atomiques associés ainsi que leurs caractérisations ;
- les caractéristiques de QdS qui seront utilisées pour l'évaluation de la QdS et des critères intrinsèque et contextuel ;
- la configuration par défaut de chaque Groupe et Sous-Groupe.

Chaque utilisateur a choisi son Groupe et classé les assemblages de Sous-Groupes disponibles. Il a exprimé ses souhaits de qualité en donnant un poids aux Sous-Groupes et des notes aux caractéristiques de QdS proposées.

En fonction de la conception de l'application, des vœux des utilisateurs et des informations décrivant le contexte, le prototype simule le fonctionnement de l'application et de la plate-forme.

Il simule l'implantation des parties locales de la plate-forme puis déploie dynamiquement l'application en utilisant les configurations par défaut. L'application va alors s'exécuter en continu jusqu'à ce qu'une reconfiguration soit nécessaire. Le prototype permet de visualiser à tout instant la note de QdS de l'application.

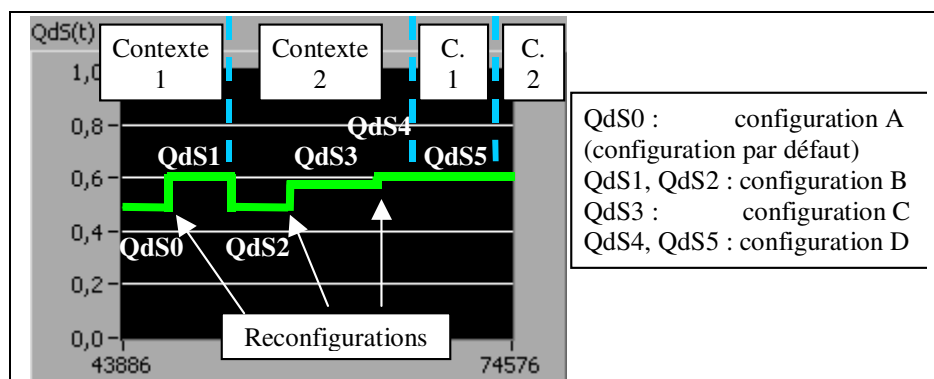
En parallèle, la plate-forme recueille les événements générés par les Processeurs Élémentaires et les Conduits. Elle identifie celui de plus haute priorité et recherche, si nécessaire, une configuration meilleure. Si elle échoue dans sa quête, elle étudie un autre événement. Dans le cas contraire, elle envoie un ordre de reconfiguration au gestionnaire de supervision et l'application est reconfigurée.

Lorsque le prototype est interrompu par l'utilisateur, l'application est arrêtée, les composants sont supprimés ainsi que les parties de la plate-forme sensées se situer sur les différents postes.

Nous donnons ici les résultats obtenus pour une application de vidéosurveillance permettant de transmettre les images et le son captés sur un site à un utilisateur distant (Laplace, 2006). Un seul Groupe appelé *Tel* correspondant à ce téléspectateur

est disponible sous un seul assemblage constitué des deux Sous-Groupes synchronisés, *Im* transmettant l'image et *So* le son. Pour transmettre le son, le Sous-Grouppe *So* propose deux décompositions fonctionnelles avec ou sans compression des données lors de la transmission. Pour transmettre les images, le Sous-Grouppe *Im* propose six décompositions fonctionnelles qui diffèrent soit par la présence de compression ou de modification des images soit par l'ordre des traitements appliqués aux images. Enfin un composant utilisé dans l'application peut être remplacé par un composant jouant le même rôle mais offrant une qualité différente. Cette application propose alors 135 configurations possibles.

L'un des tests effectués (Figure 11) a permis de caractériser le comportement de la plate-forme et de l'application lorsque le contexte d'exécution oscille. L'application est tout d'abord déployée dans un contexte favorable — contexte 1, C.1 — où ni les postes ni le réseau ne sont saturés puis ce contexte subit une dégradation — contexte 2, C.2 — due à la saturation de l'un des postes. Il revient ensuite à son état antérieur — contexte 1 — avant de se dégrader à nouveau — contexte 2.

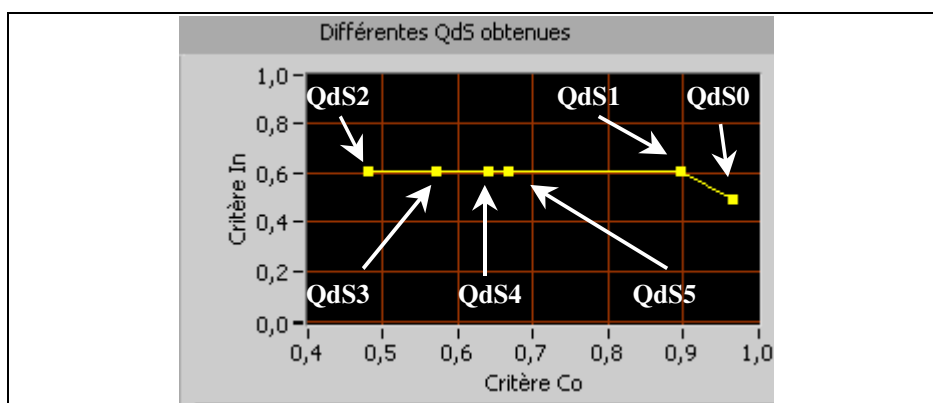


**Figure 11.** Évolution de la QdS lors d'oscillations du contexte

La première reconfiguration correspond à une amélioration du déploiement — configuration par défaut — et consiste à remplacer un composant par un autre plus performant. La deuxième reconfiguration intervient après une dégradation du contexte et consiste à déplacer un composant du poste saturé vers un autre poste. La troisième reconfiguration améliore le résultat ainsi obtenu par le déplacement d'un autre composant vers un poste non saturé.

Nous remarquons que, suite à cette troisième reconfiguration, lorsque le contexte se dégrade à nouveau — passage du contexte 1 au contexte 2 —, la QdS ne varie pas et la plate-forme ne reconfigure plus l'application. Notons que ce ne serait pas le cas si dans la phase précédente la plate-forme avait optimisé les deux critères,

intrinsèque et contextuel, et avait proposé d'atteindre QdS1 par une reconfiguration. Comme elle s'est contentée de QdS4 qui a moins de potentiel (Figure 12) mais fournit le même service du point de vue de l'utilisateur, aucune reconfiguration n'est nécessaire lorsque le contexte continue à osciller. L'imprédictibilité des variations du contexte et le souhait de maintenir la plasticité de l'application justifie donc a posteriori l'algorithme utilisé par la plate-forme.



**Figure 12.** Evolution des critères de QdS lors d'oscillations du contexte

Ce test est particulièrement intéressant car il permet de mettre en évidence que, lorsque le contexte retrouve sa situation antérieure après avoir évolué, la plate-forme propose une QdS de note identique mais en utilisant une configuration différente. La note de QdS reflétant l'adéquation entre le service fourni et celui souhaité par l'utilisateur, la configuration choisie, bien que différente, fournit donc une satisfaction équivalente à l'utilisateur. Ceci illustre bien le fait que seule la satisfaction de l'utilisateur a de l'importance. De plus, en ne cherchant pas l'optimum, la plate-forme Kalinahia stabilise le service et donne une plasticité meilleure et donc un comportement meilleur à l'application illustrant ainsi une notion proche du dilemme stabilité/précision connu en automatique. Une recherche exhaustive de la meilleure configuration — si elle était possible — ne présenterait pas cette plasticité ni ce pouvoir stabilisant.

## 6. Conclusion

En proposant de guider la conception, la réalisation et l'exécution des applications multimédia réparties vers l'obtention de la meilleure QdS possible pour les utilisateurs, nous espérons que nos travaux apportent une réponse aux problèmes de gestion de la QdS que les méthodes habituelles ne peuvent résoudre en particulier

dans des environnements dépourvus de mécanismes de réservation de ressources. Ainsi nous avons tout d'abord élaboré des modèles cohérents de QdS et d'application, puis une méthode de conception reposant sur ces modèles et enfin Kalinahia, une plate-forme d'exécution, que nous avons validée par un simulateur. L'objectif est en permanence de fournir le service le plus satisfaisant possible aux utilisateurs en évitant les variations brusques et non plus d'optimiser des critères techniques de qualité — CALiF, JQoS, QuO.

La plate-forme Kalinahia que nous proposons réalise ainsi une adaptation particulièrement efficace de l'application au contexte car elle est plus dynamique encore que ce qui est souvent présenté — CALiF, Agilos — puisque les politiques d'adaptation sont déterminées en cours d'exécution en fonction de la proximité de service. De plus, elle concerne à la fois la structure, les fonctionnalités et l'ordonnancement ce qui la rend plus complète que les systèmes habituels — JQoS, Agilos. Elle traite aussi bien des améliorations que des dégradations du contexte. En effet, même si son objectif demeure à tout instant d'améliorer la QdS telle que nous l'avons définie, en pratique, ceci peut passer par une dégradation des caractéristiques intrinsèques du service — taille des images, nombre de couleurs, etc. — de manière à obtenir une amélioration de la qualité globale grâce à un compromis entre les critères intrinsèque et contextuel. Cette possibilité d'évolution dans les deux sens de la qualité constitue donc un progrès par rapport aux systèmes habituellement proposés tels que JQoS où seule la dégradation est réalisée de façon automatique. Enfin, grâce à l'heuristique utilisée, la plate-forme Kalinahia propose une solution à un problème NP-complet non pas à partir d'une vision uniquement mathématique — comme le sont les solutions issues de l'étude des graphes pour les réseaux — mais en considérant l'utilisateur puisqu'elle se base sur la proximité de service.

D'autre part, nous pensons que plate-forme et application doivent être intimement liées depuis la phase de conception jusqu'à celle de l'exploitation afin de ne laisser à l'application que les aspects métier et de permettre ainsi la réutilisation de composants ou l'emploi de composants sur étagère — COTS — ce qui n'est pas toujours pris en compte par les travaux sur la QdS — QuO. C'est pourquoi nous souhaitons réaliser non plus uniquement une plate-forme d'exécution mais également une aide à la conception. Pour cela, il sera vraisemblablement nécessaire d'étudier une taxonomie des rôles des composants de manière à automatiser la méthode de conception et l'intégration de composants en cours d'exécution.

De plus, lorsque les travaux de notre équipe concernant la modélisation des composants logiciels et des flux pour le multimédia auront abouti, nous pourrons les utiliser pour implanter notre modèle de plate-forme. Nous pourrons alors proposer une gestion de la synchronisation intégrée à la QdS alors que nos modèles actuels la posent en condition sine qua non à toute transmission. La plate-forme disposera ainsi d'un degré de liberté supplémentaire pour optimiser la QdS.

Enfin, il sera intéressant d'étudier la possibilité d'inclure notre plate-forme dans un système plus ambitieux permettant d'utiliser l'adaptation telle que nous la

proposons dans tous les cas de figure mais également d'y adjoindre l'exploitation des garanties de service lorsque cela est possible. Nous pourrions alors utiliser une adaptation du réseau concernant les serveurs et utilisant des nœuds actifs. Ainsi le service sera optimisé du point de vue de l'utilisateur, selon notre objectif, mais aussi du point de vue du fournisseur, préoccupation à l'origine de la notion de QoS.

## 6. Bibliographie

- Ayed D., Taconet C., Bernard G., « Architecture à base de composants pour le déploiement adaptatif des applications multicomposants », *Actes des Journées Composants 2004*, Lille, France, Mars 2004.
- Comité Consultatif International Télégraphique et Téléphonique, CCITT, Rec.I.350, General Aspects of Quality of Service and Network Performance in Digital Networks, International Telecommunication Union, Geneva, 1989.
- Coutaz J., Nigay L., « Architecture logicielle conceptuelle des systèmes interactifs », *Analyse et conception de l'IHM*, Hermès 2001 Chapitre 7.
- Dalmou M., Roose P., Bouix E., Luthon F., « A Multimedia Oriented Component Model », *Proc. of the IEEE 19th International Conference on Advanced Information Networking and Applications*, Tamkang University, Taiwan, March 2005.
- DaSilva L.A., « Pricing for QoS-enabled Networks: A Survey », *IEEE Communication Surveys and Tutorials*, vol. 3, no. 2, Second Quarter 2000, pp. 2-8.
- Eles P., Doboli A., Pop P., Peng Z., « Scheduling with Bus Access Optimization for Distributed Embedded Systems », *IEEE Transactions on VLSI Systems*, volume 8, N°5, p. 472-491, October 2000.
- Firesmith D.G., « Using Quality Models to Engineer Quality Requirements », *Journal of Object Technology (JOT)*, 2 (5), Swiss Federal Institute of Technology, Zurich, Switzerland, p. 67-75, September/October 2003.
- Garcia É., Lapayre J.-C., Renard F., Ba T., « CAliF multimédia : une plate-forme à objets pour le développement de télé-applications multimédia », *Réseaux et Systèmes Répartis Calculateurs Parallèles*, n° sp. Télé-Applications, Hermès, vol.13, n°2, mars 2001, p 295.
- Garcia É., Une plate-forme de développement pour applications coopératives multimédia intégrant la gestion de la qualité de service, Thèse de l'Université de Franche-Comté, novembre 2001.
- Garey M. R., Johnson D. S., *Computers and Interactability: A guide to the theory of NP-completeness*, W. H. Freeman and Company, San Francisco, 1979.
- Gu X., Nahrstedt K., Yuan W., Wichadakul D., XU D., « An XML-based QoS Enabling Language for the Web », *Journal of Visual Language and Computing*, special issue on Multimedia Languages for the Web, vol. 13, num. 1, pp. 61-95, Academic Press, 2002.
- Hafid A., von Bochmann G., Dssouli R. « Distributed Multimedia Application and Quality of Service : A Review », *Electronic Journal on Networks and Distributed Processing*, N°6, 1998, p 1-50.

- Kuipers F., van Mieghem P., « MAMCRA: a constrained-based multicast routing algorithm », *Proc. of Computer Communications*, vol. 25, pp.802-811, 2002.
- Laplace M. S., Conception d'architectures logicielles pour intégrer la qualité de service dans les applications multimédias réparties, Thèse de l'Université de Pau et des Pays de l'Adour, 11 mai 2006.
- OSGi™ Alliance- The Dynamic Module System for Java - *About the OSGi Service Platform* - Technical Whitepaper, June 2007.
- Li B., Kalter W., Nahrstedt K., « A Hierarchical Quality of Service Control Architecture for Configurable Multimedia Applications », *Journal of High Speed Networks*, Special Issue on Management of Multimedia Networking, IOS Press, Vol. 9, pp. 153-174, 2001.
- Parkin M., Fluet C-D, Bade R., *Introduction à la microéconomie moderne*, Editions ERPI, 1992, ISBN 2-7613-0673-2.
- Rakotonirainy A., Indulska J., Loke S., Zaslavsky A., «Middleware for Reactive Components : An Integrated Use of Context, Roles, and Event Based Coordination », in *Proc. of the IFIP/ACM International Conference on Distributed Systems Platforms*, Heidelberg, Allemagne, Novembre 2001.
- Schantz R.E., Loyall J.P., Rodrigues C., Schmidt D.C., Krishnamurthy Y., «Flexible and Adaptive QoS Control for Distributed Real-time and Embedded Middleware ». *Proc. of the ACM/IFIP/USENIX International Middleware Conf.*, Rio de Janeiro, Brazil, June 2003.
- Stephen S., Yu W., Fariaz K., «Development of Situation-Aware Application Software for Ubiquitous Computing Environments », in *Proc.of COMPSAC'02*, Angleterre, 2002.
- Vogel A., Kerherve B., von Bochmann G., Gecei J., « Distributed multimedia applications and Quality of Service :- A Survey-» *IEEE Multimedia Journal*, august 1995.
- Wang N., Gill C., Schmidt D., Gokhale A., Natarajan B., Loyall J., Schantz R., Rodrigues C., «QoS-enabled Middleware ». Chapter in *Middleware for Communications*, Qusay H. Mahmoud (Editor), Wiley, July 2004.
- Zhu W., Georganas N. «JQoS : Design and Implementation of a QoS-based Internet Videoconferencing System using Java Media Framework (JMF) » *Proc. of CCECE'2001*, Canadian Conference on Electrical and Computer Engineering, Toronto, Canada.