

## Rapport de Recherche



### Gestion prédictive de la qualité de service

**BELKHIR Abdelkader,**  
belkhir@wissal.dz,

**BOUYAKOUB Fayçal  
M'hamed,**  
bouyakoub\_fm@hotmail.com

LSI-TR-1004

Juin 2004

FACULTE ELECTRONIQUE & INFORMATIQUE  
Département informatique  
El Alia BP n°32 Bab Ezzouar 16111 Alger.  
Tél / Fax : 213 (0) 21 24 79 17 - 24 76 07

# Gestion prédictive de la qualité de service

**BELKHIR Abdelkader,**  
belkhir@wissal.dz,

**BOUYAKOUB Fayçal M'hamed,**  
bouyakoub\_fm@hotmail.com

USTHB, Faculté Electronique et Informatique  
Département Informatique BP 32 El-Alia Alger ALGERIE

**Résumé :** Plusieurs solutions du problème QoS ont été développées, tant au niveau des routeurs qu'au niveau des protocoles, mais elles agissent toutes aux couches les plus basses du modèle OSI. Notre solution, contrairement aux précédentes, agit au niveau application, ce qui lui permet d'exploiter au mieux les caractéristiques du langage SMIL : prédiction du scénario futur, possibilité d'effectuer des préchargements. La solution proposée dans cet article s'appuie sur la technique de préchargement et de streaming afin d'améliorer la qualité de service pour des présentations au format SMIL.

**Mots clés :** préchargement, streaming, SMIL, QoS,

## 1. Introduction

L'évolution des technologies réseaux ainsi que l'accroissement de leurs performances ont fait que l'Internet est devenu une infrastructure viable pour supporter les services multimédias [1]. Ces derniers concernent tous les aspects de la vie moderne: la communication personne à personne, la visioconférence, les présentations de documents multimédias synchronisés, l'enseignement et la formation à distance,...etc. Ces catégories d'applications sont gourmandes en ressources : capacité de stockage, vitesse d'exécution et consommation de bande passante. La disponibilité de machine ayant une grande capacité de stockage de l'ordre de giga-octet ainsi qu'une vitesse d'exécution très appréciable résout en partie la problématique de ressources. Cependant, toutes les solutions ne peuvent pas toujours anticiper l'effet de la congestion sur le client.

Pour cela, la gestion de la bande passante mérite une attention particulière. Il s'agit de dépasser la stratégie *Best-effort* afin d'aboutir à une stratégie exploitant les caractéristiques de l'application : c'est la qualité de service au niveau application. Les classes d'application à considérer sont les systèmes de présentation multimédia intégrant le son, l'image, la vidéo et le texte. A cet effet, il y a lieu de définir des solutions adaptées à la nature des applications multimédias. Elles sont caractérisées par des flux multimédias dont la rupture même momentanée a un impact direct sur la qualité de service.

Le format de spécification utilisé par l'environnement auteur joue un rôle important dans la réalisation d'un système de présentation multimédia, puisque la popularité du langage choisi affecte directement les possibilités de diffusion de ces présentations à grande échelle, notamment sur le Web. Ce qui peut justifier le choix du langage HTML. Cependant, l'absence de mécanismes de synchronisation fait que HTML est inadapté à la modélisation de présentations multimédias synchronisées où la composante temporelle est prédominante. L'interface papier [2] met en jeu plusieurs flux multimédias synchronisés, c'est évidemment cet aspect de synchronisation qui justifie le choix du langage SMIL [3] langage de spécification des présentations multimédia. De plus, le langage SMIL est un standard du Web. Il offre la possibilité de décrire l'organisation spatiale et temporelle d'une présentation grâce à un ensemble de modules. Parmi ses fonctionnalités, on retrouve le mécanisme de préchargement (l'élément *prefetch*) qui indique qu'un élément media sera utilisé dans le futur. Les deux mécanismes de préchargement et de streaming constituent la pierre angulaire de notre contribution à travers cet article.

Dans la section 2, on décrit les motivations et choix de notre solution ainsi qu'un aperçu sur des travaux similaires. La section 3 présente notre solution de gestion prédictive de la qualité de service en utilisant un procédé statique. Les résultats obtenus nous amène à développer davantage la solution en adoptant une approche dynamique. La section 4 décrit la contribution d'approche dynamique qui est complémentaire à l'approche statique. On terminera l'article par une discussion des résultats obtenus et conclusion sur le travail accompli.

## 2. Motivations

Depuis l'apparition du problème QoS, plusieurs solutions ont été développées [4] [5] [6] [7], tant au niveau des routeurs qu'au niveau des protocoles. Cependant, mais elles agissent toutes aux couches les plus basses du modèle OSI. Notre solution, agit au niveau application, ce qui lui permet d'exploiter au mieux les caractéristiques de SMIL [3] : prédiction du scénario futur, possibilité d'effectuer des préchargements ...

La qualité de service se rapporte à certaines caractéristiques d'une connexion réseau (délai, gigue, bande passante, disponibilité) et s'applique sur les nœuds reliant ses deux extrémités. La garantie d'une qualité de service peut être obtenue par [1]: la surabondance de bande passante, l'augmentation de la puissance des routeurs, l'adaptation des terminaux. En dépit de la disponibilité de stations et de routeurs plus puissants ; il en demeure que la surabondance de la bande passante nécessite un investissement à l'échelle mondiale. Afin de corriger les défauts du service *Best-effort*, le terminal s'efforce de corriger la dégradation par un procédé de remplacement des paquets perdus ou en retard. Néanmoins, cette correction nécessite un temps de calcul et s'avère inefficace si elle dépasse quelques millisecondes. Les routeurs à leurs tours peuvent agir sur la qualité de service en adoptant les architectures *IntServ* ou *DiffServ* [8].

Afin d'anticiper l'effet de la congestion sur le client, il y a lieu de définir des solutions adaptées à la nature des applications multimédias. Elles sont caractérisées par des flux multimédias dont la rupture même momentanée a un impact direct sur la qualité de service.

L'adoption d'une approche de niveau application est motivée par deux éléments essentiels:

- a. Une approche de qualité de services sur la couche application complète les approches sur les couches inférieures et profite des améliorations offertes par ces dernières.
- b. SMIL est un langage de niveau application. Une approche de qualité de service au niveau applicatif s'adapte avec la sémantique de ce langage.

Le choix de la technique du préchargement est fortement lié au choix du langage SMIL sur lequel se sont basés nos travaux. Le préchargement a été introduit dans *SMIL 2.0* mais de manière élémentaire. Il permet de mieux exploiter les ressources disponibles sans affecter ou modifier l'architecture réseau.

Des solutions ont été élaborées afin d'assurer l'indépendance de la présentation vis-à-vis des variations de la bande passante, en réduisant les délais d'accès par chargement des médias avant la date nominale de leur présentation. Ces solutions agissent au niveau application, ce qui leur permet de s'adapter à la sémantique de SMIL. Le préchargement dans le TLSA Player [9] peut s'effectuer selon deux approches différentes. L'approche statique consiste à précharger la totalité des éléments avant le début de la présentation, ce qui risque de saturer les ressources systèmes (mémoire) et réseau (bande passante) lorsqu'on a un nombre considérable de médias. Dans l'approche dynamique, le TLSA Player se contente de précharger les éléments des  $n$  niveaux suivants à partir de l'état courant. Cette approche réduit la consommation des ressources, mais elle peut converger vers l'approche statique dans le cas où le niveau de préchargement est assez grand. De plus, la configuration du niveau de préchargement par des utilisateurs inexpérimentés peut causer une pénurie de ressources. La deuxième solution, présentée dans [10], tente de réduire les délais d'accès en calculant l'instant idéal pour lancer les préchargements, afin d'assurer la disponibilité des médias à temps. Cette solution présente l'avantage de ne pas saturer la mémoire, vu que le préchargement d'un objet se fait juste avant son temps de début. Cependant, l'échange de messages entre le client et les serveurs, augmente la charge du réseau. De plus, la nature instable de la bande passante peut influencer négativement sur les temps de chargement calculés des médias, induisant le non respect des échéances temporelles.

## 3. Gestion prédictive : approche statique

Le préchargement des objets multimédias dans SMIL repose sur le principe de la récupération des données par anticipation, avant leur utilisation effective dans le temps. Le préchargement doit être mis en œuvre d'une manière efficace, afin de mieux exploiter les ressources disponibles et sans affecter le scénario temporel d'exécution. Lorsqu'une requête est reçue sur le serveur hébergeant la présentation SMIL, le module de préchargement est automatiquement exécuté. Ce module génère un nouveau fichier SMIL à partir du fichier de base incluant les commandes de préchargement des objets multimédias de la présentation. Au niveau du client, l'exécution de la présentation se fera par un player supportant la version 2 de SMIL.

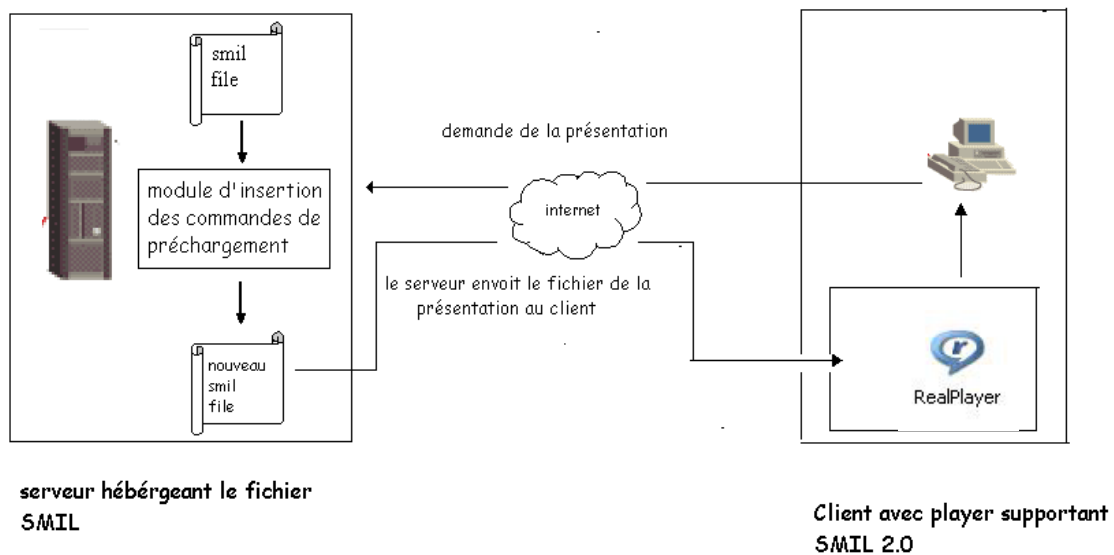


Figure 1. Architecture du système de préchargement statique

### 3.1 Modélisation du scénario temporel de la présentation: Génération de l'arbre abstrait

La production du scénario temporel d'exécution d'une présentation multimédia consiste à obtenir:

- Une modélisation des relations temporelles entre les objets multimédias de la présentation. Ces informations seront stockées dans *l'arbre abstrait* afin de décrire l'organisation logique du document en termes d'éléments composites et éléments de base mais aussi définir l'ordre de précedence entre les différents éléments.
- Les valeurs de début et de fin effectives de chaque objet multimédia participant dans le déroulement de la présentation. Ces informations seront stockées dans une structure de données appelée la *table de médias*. Au fur et à mesure de l'analyse temporelle du document SMIL, la *table de médias* est mise à jour et on obtient à la fin les temps de début et de fin effectives de chaque objet multimédia participant dans le déroulement de la présentation.

Nous utiliserons les fonctions suivantes pour la production du scénario temporel d'exécution:

La procédure *StartTag*, qui est appelée pour chaque balise ouvrante d'un élément *par*, *seq*, *prefetch* ou un objet multimédia de base. Les fonctions *GetBeginTime* et *GetEndTime*, qui servent au calcul des temps de début et de fin effectifs des objets multimédias de base. La procédure *EndTag*, qui est appelée pour chaque balise fermante d'un élément.

Le résultat final sera sous forme de deux tables:

1. La table du scénario ou "*MediaTable*": contient toutes les informations relatives aux objets multimédias
2. La table des composites ou "*TagTable*": contient les informations relatives aux objets composites. Chaque ligne correspond à un nœud de l'arbre.

La combinaison de ces deux tables, ainsi que le chaînage entre les éléments de ces dernières forme l'arbre abstrait.

### 3.2 Insertion des commandes de préchargement

Dans notre approche dite statique [11], les commandes de préchargement sont insérées dans le document source avant l'exécution de la présentation, nous distinguons deux cas:

- Le premier concerne les objets en séquence (<seq>...</seq>).
- Le deuxième concerne les objets en parallèle (<par>...</par>).

Les règles définies précédemment sont traduites par un algorithme [11] [12].

### 3.3 Affectation de la bande passante

L'autre étape consiste à déterminer la quantité de bande passante à affecter à chaque commande de préchargement insérée. Ce paramètre dépend d'une part du type de la connexion du client, et les besoins du système et des objets en cours d'exécution d'autre part.

La bande passante disponible au niveau du client est déterminée en utilisant l'attribut SMIL "*systemBitrate*". Cet attribut, combiné avec la balise *switch*, permet de tester si la bande passante disponible est supérieure ou égale à la valeur spécifiée par l'attribut "*systemBitrate*". Il suffit donc de dupliquer le code SMIL pour chaque classe de bande passante. L'ordre de classification doit être décroissant. Au niveau du client, le player SMIL sélectionne le bloc qui correspond à la bande passante disponible et exécute le code correspondant.

Des études statistiques [13], ont montré qu'environ 30% de la bande passante est réservée pour le trafic d'en-tête et la correction des erreurs. La bande passante exploitable est donc 70% de la bande passante totale. Pour déterminer la bande passante consommée par les objets multimédias, nous les classons en deux catégories:

#### a. *Les objets hébergés par un serveur Web*

Ces objets utilisent le protocole HTTP. Ils doivent être téléchargés entièrement avant d'être joués. HTTP étant un protocole sans états, la notion de flux est inexistante, il n'y a donc aucune règle permettant de définir les besoins de ces objets en termes de bande passante. Nous pouvons contourner ce problème en attribuant une quantité nécessaire de bande passante à ces objets pendant les premières secondes de leur exécution. Lorsque l'objet est entièrement téléchargé, il libère la bande passante ce qui nous permettra de l'allouer au préchargement [13].

#### b. *Les objets hébergés par un serveur de streaming*

Ces objets sont généralement des médias continus tels que l'audio et la vidéo. L'objet multimédia est récupéré et diffusé en temps réel à partir d'un serveur de streaming. Dans ce cas, il est nécessaire de garantir une bande passante égale au format de compression [13]. Par conséquent, la bande passante à allouer au préchargement d'un objet donné est obtenue par la formule suivante:

$$BP\_Prefetch(object) = 70\%BP\_Totale - \Sigma BP\_StreamObjects - \Sigma BP\_HttpObjects$$

où:

**BP\_Totale:** Bande passante totale disponible suivant le type de connexion du client

**BP\_StreamObjects:** Bande passante consommée par l'objet diffusé en streaming en cours d'exécution

**BP\_HttpObjects:** Bande passante allouée à l'objet téléchargé en HTTP en cours d'exécution

**BP\_Prefetch:** Bande passante à attribuer au préchargement de l'objet. Dans le cas d'un bloc "seq", cette quantité sera entièrement allouée au préchargement de l'objet suivant. Quant au cas d'un bloc "par", elle sera répartie sur les préchargements des éléments ayant un décalage temporel par rapport au début du bloc.

### 3.4 *Contraintes à respecter*

La commande de préchargement est un outil puissant à utiliser avec modération. Une utilisation exagérée de cette commande conduit à une dégradation de la présentation à cause de l'épuisement des ressources système (mémoire) et réseau (bande passante).

#### a. *Bande passante*

La bande passante allouée au préchargement doit tenir compte des besoins du système d'une part et des objets en cours d'exécution d'autre part. Lorsque la bande passante à allouer au préchargement est trop faible, le gain est insignifiant. Nous fixons un seuil de 2 Kbps à garantir pour effectuer les préchargements.

#### b. *Mémoire*

La technique de préchargement maintient les données en mémoire jusqu'à leur exécution. Le téléchargement de fichiers volumineux tels que l'audio et la vidéo peut rapidement saturer la mémoire. Pour éviter une telle situation, nous nous limitons au préchargement du preroll estimé à 15 secondes [13].

#### c. *Durée du préchargement*

Le préchargement n'a aucun intérêt si sa durée est trop petite. Nous fixons alors un seuil limite minimal de durée à 5 secondes.

#### d. *Respect des échéances temporelles*

L'insertion des commandes de préchargement ne doit pas affecter le scénario temporel de la présentation, les échéances temporelles des objets doivent rester inchangées.

### 3.5 *Tests et analyse des résultats*

Les documents SMIL et les objets multimédias ont été hébergés sur un serveur de streaming exécutant Real System Server [13]. On dispose pour chaque présentation de deux versions: l'une sans commandes *prefetch* et

l'autre avec des commandes *prefetch*. Pour exécuter les présentations nous avons utilisé le logiciel *RealOne Player* [13] qui offre la possibilité de visualiser un diagramme du flux de données entrant. Les tests ont permis d'obtenir les diagrammes des flux suivants.

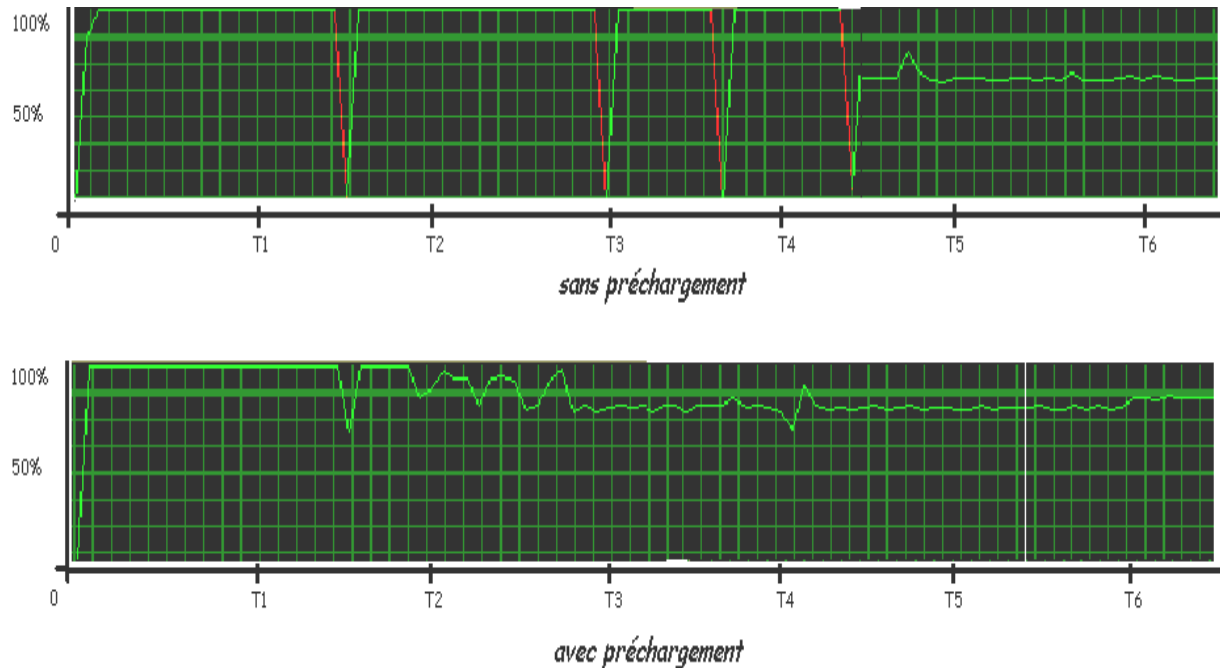


Figure 2. Flux de données obtenus à travers *RealOne Player*

La courbe en pointillés représente la présentation initiale, et la courbe en traits la présentation après l'insertion des commandes de préchargement.

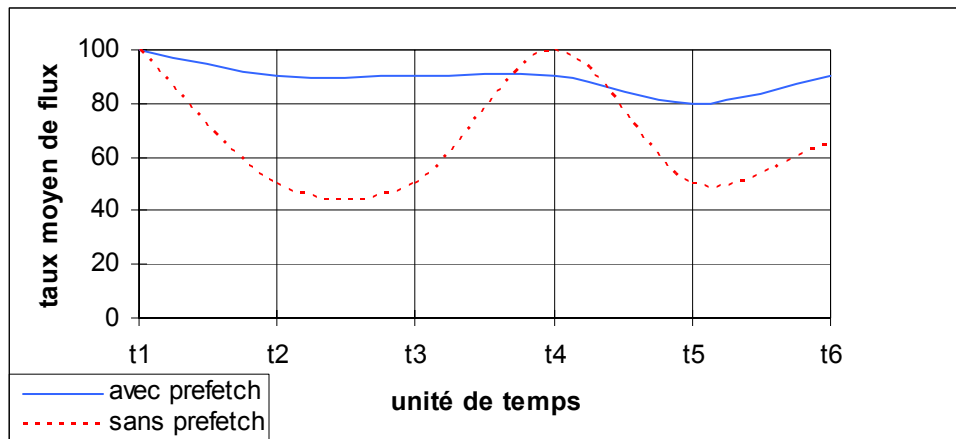


Figure 3. Flux moyen de données

L'interprétation des résultats montre que la présentation sans *prefetch* est soumise aux variations de la bande passante (cas de congestion), ce qui provoque des *rebufférisations* continues. Par contre, pour la présentation avec préchargement, la qualité de service reste satisfaisante du fait que le player ne se contente pas de charger l'objet courant, mais tire profit de la disponibilité de la bande passante pour anticiper le chargement des objets selon les commandes *prefetch* insérées dans le document SMIL. Nous remarquons que le flux moyen de données reste supérieur (de 30% en moyenne) au flux de la présentation initiale (sans préchargement).

Lorsqu'il y a congestion, nous avons constaté que ses effets tardent à se faire sentir dans le cas de la présentation avec préchargement. Ceci est dû à l'exploitation de la réserve obtenue à travers le préchargement. Ce qui permet de résister à la chute de bande passante en attendant qu'elle retrouve un taux acceptable. En cas de congestion prolongée, la réserve s'épuise et on se retrouve dans le cas d'une présentation sans préchargement. De plus, la bande passante n'est pas fixe, sa valeur fluctue dans le temps, or la méthode de préchargement statique insère les commandes de préchargement dans le source du document SMIL "statiquement" avant le début de la

présentation. Par conséquent, une surabondance de la bande passante en un moment donné de la présentation ne pourra être pleinement exploitée pour effectuer des préchargements car cette situation ne peut être connue à priori qu'au moment de la présentation. Les résultats satisfaisants obtenus par l'approche statique nous ont encouragés à étendre cette solution en passant à l'approche dynamique qui devrait améliorer davantage la qualité de service. La solution consiste à effectuer les préchargements de manière dynamique au moment de la présentation, les valeurs de bande passante étant plus précises, ceci permettrait une meilleure gestion des préchargements.

#### 4. Gestion prédictive : approche dynamique

L'approche dynamique permet de compléter l'approche statique en effectuant des préchargement à la volée, i.e. pendant l'exécution de la présentation. Pour cela, on a intégré notre solution au sein du player SMIL.

Notre solution passe par deux étapes:

1. Insertion des balises <ref /> et création des fichiers de préchargement
2. Remplissage des fichiers de préchargement *PrefetchFile.smi* à la volée

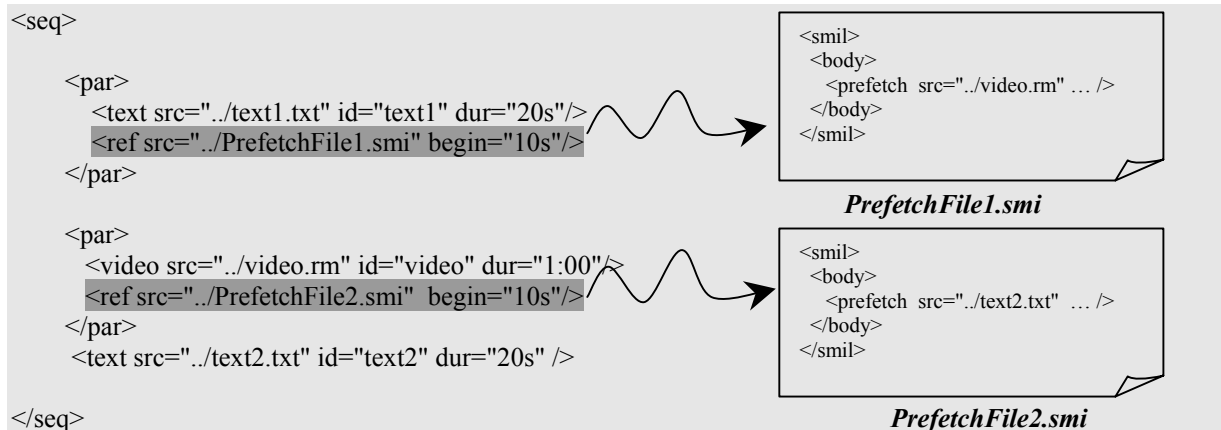
Cette solution se base, tout comme l'approche statique, sur l'arbre abstrait afin de récupérer les informations de synchronisation de la présentation SMIL. L'insertion des balises de référence se fait automatiquement au début de la présentation aux endroits où les opérations de préchargement sont possibles. On distingue deux cas:

- a. **Cas d'un groupe "seq"**: Dans le cas des éléments en séquentiel, on insère la balise de référence en parallèle avec l'élément en cours. Cette balise fait référence à un fichier SMIL appelé *PrefetchFile.smi* et généré à l'insertion de la balise <ref />. Ce fichier contient une balise <prefetch /> de l'élément qui suit l'élément courant.

exemple:

```
<seq>
  <text src="../text1.txt" id="text1" dur="20s"/>
  <video src="../video.rm" id="video" dur="1:00"/>
  <text src="../text2.txt" id="text2" dur="20s" />
</seq>
```

Le code SMIL obtenu après insertion des balises de référence est:



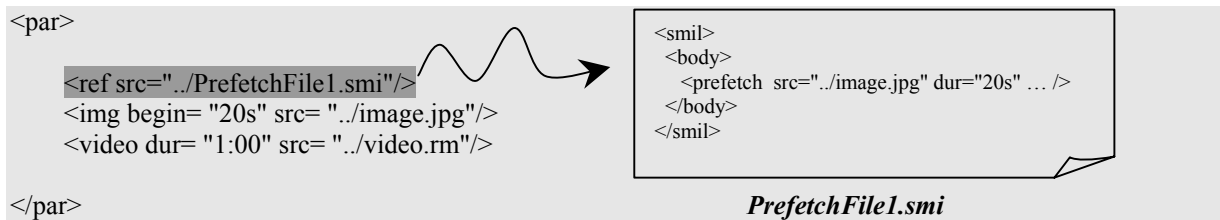
- b. **Cas d'un groupe "par"**: Dans ce deuxième cas de figure, on a deux possibilités:

b1. **Cas des décalages à l'intérieur du groupe "par"**: dans cette situation, on profitera de ces décalages pour insérer une balise <ref /> à l'intérieur du groupe "par". Cette balise fait référence à un fichier SMIL qui contient le préchargement de l'objet ou des objets décalé(s) par rapport au début du père.

Prenons l'exemple suivant:

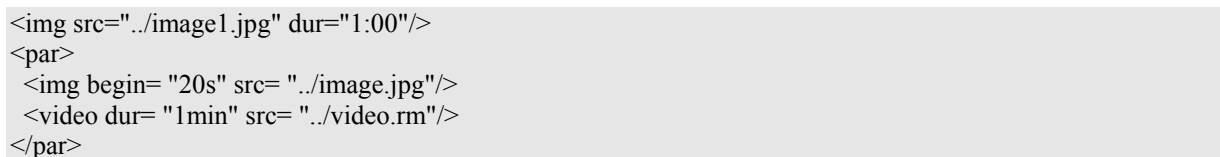
```
<par>
  
  < video dur="1:00src=" ../video.rm"/>
</par>
```

Le code SMIL obtenu après l'insertion des balises de référencement est le suivant:

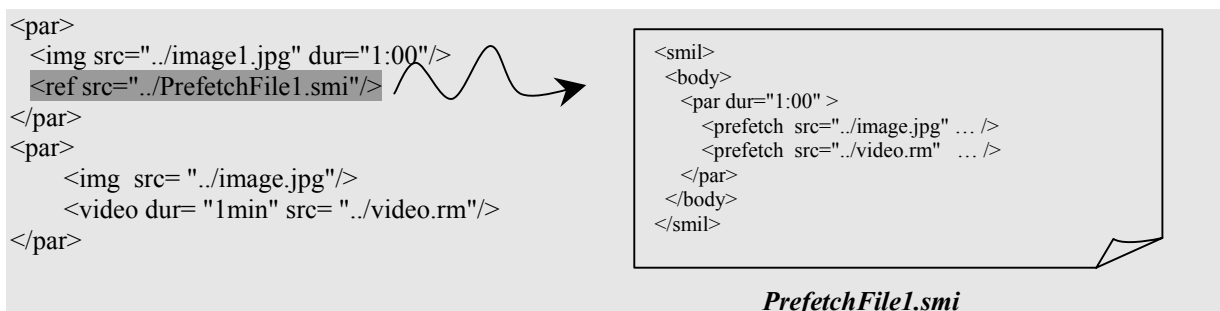


**b2. Cas d'un élément multimédia frère d'un groupe "par":** c'est le cas où juste avant le "par", on a un élément appartenant à un "seq". Dans ce cas, si on dispose d'une quantité suffisante de bande passante pour effectuer des préchargements sur un ou plusieurs objets du "par", alors on insère en parallèle à l'élément courant une balise <ref/> qui fait référence à un fichier de préchargement contenant un groupe parallèle de commandes de préchargement des objets commençant au début du "par".

Voyons l'exemple suivant:



Le code SMIL obtenu après l'insertion des balises de référencement est le suivant:



Dans le cas d'éléments streamés, le décalage permet au player de précharger le preroll de cet objet s'il n'a pas été préchargé auparavant. Par la suite, la bande passante allouée au préchargement sera la bande passante restante puisque l'objet streamé consommera de la bande passante pendant son exécution selon le format de codage.

#### 4.1 Affectation de la bande passante

L'approche dynamique constitue une contribution à l'approche statique. En effet, l'approche statique ne permettait pas de connaître la valeur instantanée de la bande passante disponible au niveau du client. Elle procédait par tranche de type de connexion (entre 28 et 56, entre 56 et 128 Kbps...). De plus l'insertion des commandes de préchargement se faisait à la création de la présentation, ce qui nous donnait des résultats approximatifs pour les paramètres de préchargement.

Dans l'approche dynamique, le calcul de la bande passante à affecter au préchargement est donné par la formule suivante:

$$\text{AvailableBW} = 70\% * m\_RealOne.GetConnectionBandwidth() - m\_RealOne.GetBandwidthCurrent()$$

*AvailableBW* : bande passante à affecter aux préchargements

*m\_RealOne.GetConnectionBandwidth()* : bande passante de la connexion

*m\_RealOne.GetBandwidthCurrent()* : bande passante courante

#### 4.2 Fonctionnement

Cette solution est une amélioration de l'approche statique. Les procédures *StartTag()*, *EndTag()* ainsi que les procédures de calculs des temps de début et de fin *GetBeginTime()* et *GetEndTime()* utilisées dans l'approche statique seront réutilisées. Cependant, de nouvelles fonctions ont été ajoutées: la procédure d'insertion des balises de référencement *InsertRef()*, la procédure de création des fichiers *PrefetchFile.smi* ainsi que la procédure de remplissage de ces fichiers. L'utilisation d'un timer s'avère nécessaire dans notre puzzle. En effet, ce timer nous permettra de synchroniser notre application avec l'état d'avancement de la présentation afin de signaler l'arrivée

de l'instant où on doit remplir un fichier *PrefetchFile*. Le timer est armé après le remplissage de chaque fichier de préchargement afin de signaler le prochain instant de remplissage. Cette valeur est calculée suivant la formule suivante:

$$\text{PrefetchingInstant} = \text{ObjectBeginningTime} - m\_RealOne.GetPosition() - \Delta t$$

où:

**PrefetchingInstant**: c'est l'instant de remplissage du fichier de préchargement

**ObjectBeginningTime**: c'est le temps de début du média à précharger

**m\_RealOne.GetPosition()**: c'est le temps courant de la présentation

$\Delta t$ : c'est l'intervalle de temps nécessaire pour préparer le fichier de préchargement avant le lancement de l'objet préchargé. Cette valeur varie selon la vitesse de calcul de la machine.

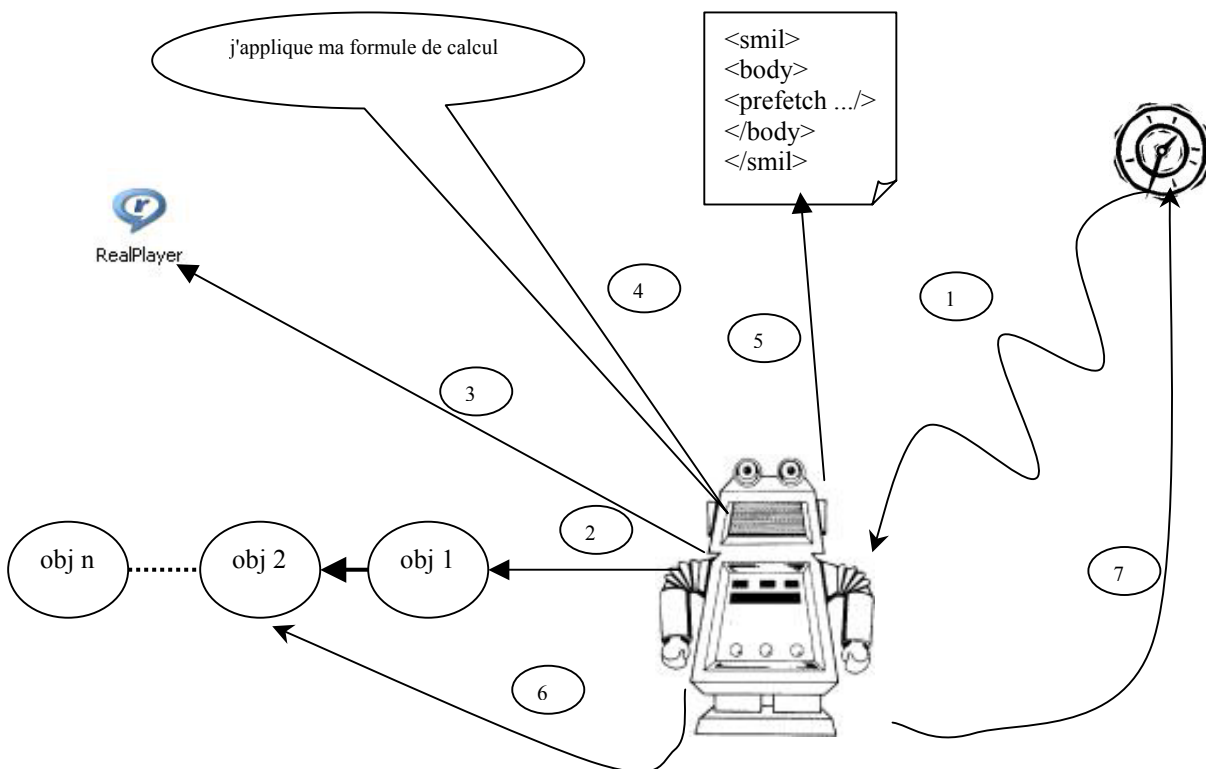


Figure 4. Fonctionnement de l'approche dynamique

- 1: signal d'horloge indiquant l'arrivée du temps de remplissage du fichier de préchargement
- 2: je récupère les informations nécessaires concernant l'objet à précharger à partir de *PrefetchInfosList*
- 3: je récupère à partir des méthodes de RealOne Player la bande passante disponible
- 4: j'applique la formule de calcul définie en section 4.1
- 5: je crée mon fichier de préchargement
- 6: je récupère à partir de *PrefetchInfosList* le temps de début du prochain élément
- 7: j'arme mon timer avec la valeur calculée par la formule définie en section 4.2

### Remarques

- *CalculateAvailableBW()* calcule la bande passante restante, *InsertPrefetchCommand()* insère les commandes de préchargements dans le fichier associé, *CalculateNextTime()* calcule le temps de la prochaine insertion des balises prefetch.

- Seul 85% de la bande passante disponible sera affecté au préchargement. nous réservons une marge de 15% afin d'éviter la saturation de la bande passante en situation de congestion [13].

#### 4.3 Maintien de la synchronisation entre l'application et la présentation

Afin de maintenir la synchronisation entre l'application et la présentation, on a utilisé les messages de retour *Callbacks* (message envoyé suite à un événement) de RealOne. En effet, RealOne offre une série de *Callbacks* qui nous permettent de suivre l'état du player. On s'est intéressé plus particulièrement à *OnPlayStateChange(NewState)*. Ce message est envoyé par le player suite à un changement d'état dans la présentation. Le nouvel état est exprimé par le paramètre *NewState*. Ce paramètre admet plusieurs valeurs:

Valeur du paramètre	Etat du player
0	stopped
1	contacting
2	buffering
3	playing
4	seeking

Tableau 1. Valeurs et états du player

Les deux états qui nous intéressent sont: buffering et playing. Chaque état s'accompagne avec des actions. Quand le player est dans un état de *rebufferisation*, l'action à effectuer c'est d'arrêter le timer de l'application. Dès le changement d'état du timer (à la reprise de la diffusion de la présentation), la player envoie le Callback avec comme paramètre *playing*. Dans ce cas, on n'a qu'à réarmer le timer avec une nouvelle valeur calculée par la fonction *CalculateNextTime()*.

#### 4.4 Implémentation et tests

Des tests ont été effectués sur le réseau du CERIST<sup>1</sup>. Les documents SMIL, ainsi que les objets multimédias ont été hébergés sur un serveur supportant le streaming. Pour jouer les présentations, nous avons utilisé le logiciel RealOne Player [13] qui offre la possibilité de visualiser le flux de données entrant. Pour simuler l'environnement d'Internet, nous avons introduit une charge supplémentaire sur le réseau.

Chaque document SMIL a été soumis au module d'insertion des commandes de préchargement (approche statique et dynamique). Les deux présentations ont été jouées dans des conditions semblables pour pouvoir les comparer. Les résultats obtenus sont représentés par les diagrammes suivants:

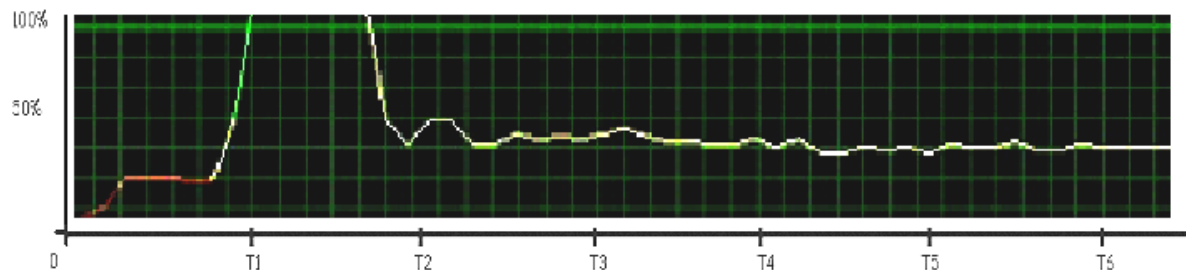


Figure 5. Flux de données obtenus à travers RealOne Player : Préchargement statique

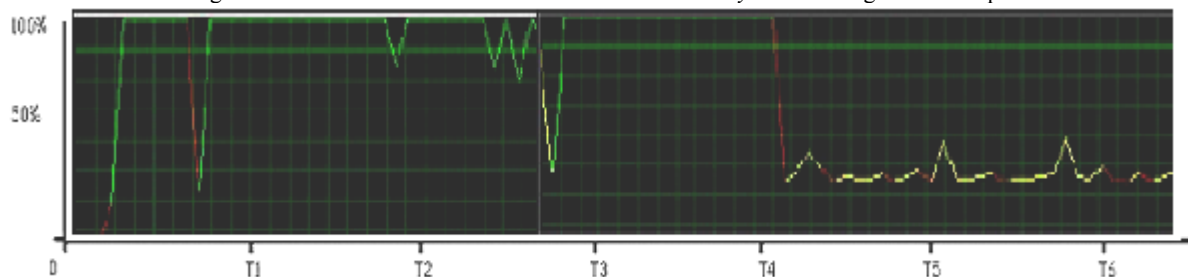


Figure 6. Flux de données obtenus à travers RealOne Player : préchargement dynamique

<sup>1</sup> Centre de recherche en information scientifique et technique : [www.cerist.dz](http://www.cerist.dz)

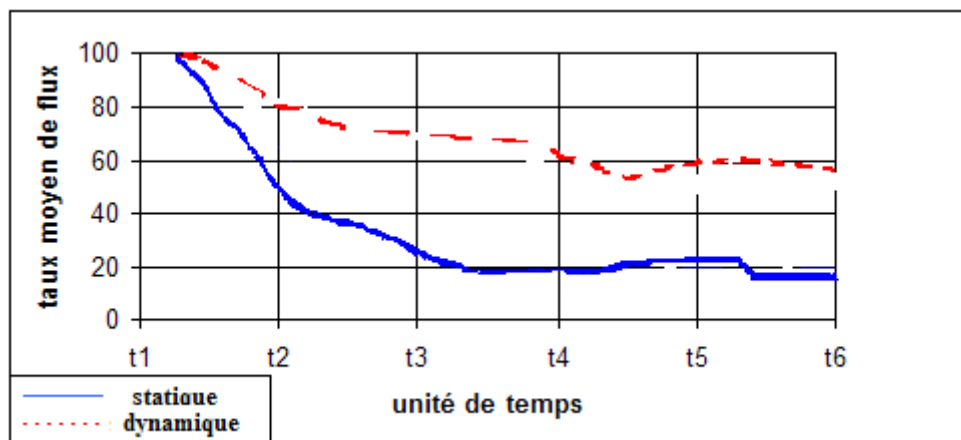


Figure 7. Flux moyen de données (statique, dynamique)

## 5. Conclusion

L'approche statique permet d'améliorer la disponibilité des médias en exécutant les commandes de préchargement insérées avant le lancement de la présentation. Cependant, cette solution présente quelques limitations. En effet, les commandes de préchargement étant insérées statiquement avant le début de la présentation, une surabondance de la bande passante en un moment donné de la présentation ne pourra être pleinement exploitée pour effectuer des préchargements, car cette situation ne peut être connue à priori qu'au moment de la présentation. Cette situation n'est pas prévisible dans le cas d'une approche statique. La solution de préchargement dynamique dispose d'un mécanisme de gestion de la bande passante en temps réel, lui permettant une exploitation optimale de cette ressource. De manière générale, les tests effectués ont montré que le mécanisme de préchargement dynamique améliore davantage (environ 35%) la qualité des présentations SMIL en permettant d'obtenir des présentations plus fiables et plus résistantes aux perturbations de la bande passante, grâce à un mécanisme temps réel de gestion de la bande passante.

En cas de congestion prolongée, l'impact sur la présentation est retardé, mais une congestion prolongée induit une chute du niveau de flux de données, menant à l'arrêt de la présentation.

La solution proposée est basée sur une approche prédictive, elle tire profit de la possibilité d'avoir le scénario temporel de la présentation à l'avance pour anticiper le chargement des médias (en utilisant les commandes prefetch de SMIL 2.0) lorsque la bande passante est abondante, et disposer ainsi d'une réserve locale en prévision d'une situation de congestion.

## 6. Bibliographie

- [1] Susbielle.J.F, "*Internet, multimédia et temps réel*", édition Eyrolles, 2001.
- [2] Belkhir.A, Bouyakoub.F.M, Smail.S, Interface papier pour des présentations multimédia, rapport interne LSI LSI-TR-0904, Juin 2004.
- [3] SMIL, "*Smil: Synchronized multimedia integration language*", url: <http://www.w3.org/AudioVideo>, 2002.
- [4] Diaz.M, Drira.K, Lozes.A et Chassot.M, "*Definition and Representation of the Quality of Service for Multimedia Systems*", 6th International Conference on High Speed Networking, HPN'95, Palma de Mallorca Balearic Islands., Spain, Septembre 11-15, 1995.
- [5] Horlait.E, Rouhana.N, "*Qualité de service dans l'architecture TCP/IP*", université Pierre et Marie Curie, Laboratoire LIP6, 2000.
- [6] Vogel.A, Kerherve.B, Bochmann.G et Gecsei.J, "*Distributed Multimedia Applications and Quality of Service: A Survey*", IEEE Multimedia Journal, Août 1995.

- [7] Poultan.A, Clayton.P, Guillarmod.J, "*The design of a bandwidth management and pricing proxy*", Department of computer science, Rhodes University, 1999.
- [8] Blake.S, Black.D, Carlson.M, Davies.E, Wang.Z, Wies.W, "*An Architecture for Differentiated services*", RFC 2475, 1998.
- [9] Sampaio.M.P.N, "*Conception formelle de documents multimédia interactifs: une approche s'appuyant sur RT-LOTOS*", thèse de Doctorat, université Paul Sabatier de Toulouse, Avril 2003.
- [10] Yang.C.C, Yang.Y.Z, "*Design of the data-retrieving engine for distributed multimedia presentations*", Multimedia and Communications Laboratory, Department of Computer Science and Information Engineering, National Chi-Nan University, Taiwan.
- [11] Belkhir.A, Nouasri.A, Bouyakoub.F.M, Smail.S, "*Predictive QoS management for SMIL presentations*", 4<sup>th</sup> international conference on Information Technology based Higher Education and Training, ITHET03, Marrakech, Morocco, Juillet 7-9, 2003.
- [12] Belkhir.A, Nouasri.A, Bouyakoub.F.M, Smail.S, "*Gestion prédictive de la qualité de service pour les présentations multimédia au format SMIL*", VI<sup>th</sup> International Symposium on Programming and Systems, ISPS 2003, Alger, Algérie, Mai 5-7, 2003.
- [13] RealNetworks, url: <http://www.realnetworks.com>, Janvier 2003.