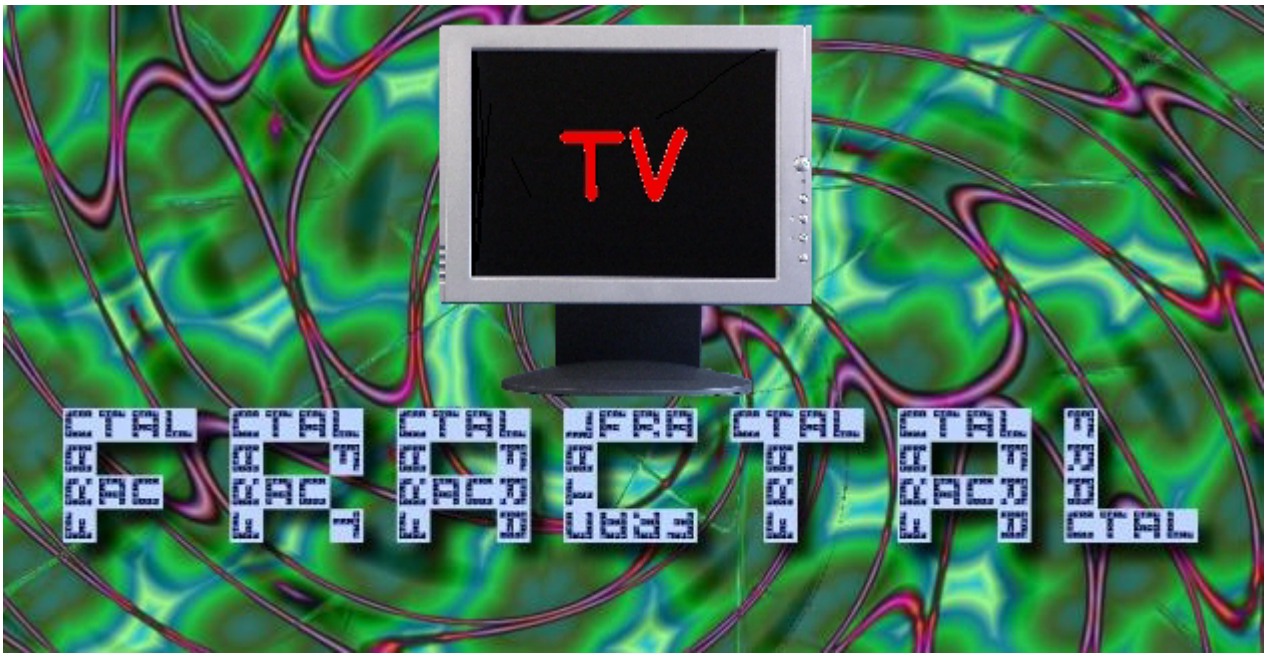
Travail d'Etude et de Recherche 2005

Le rapport



Les Membres

Encadrant	Philippe Collet
Participants	Barbier Thomas
	Castillejos Nicolas
	Nicolas Yohann
	Salageanu Emil
	Sauvan Bastien

I . Présentation des intervenants du projet

Philippe Collet

Chercheur au laboratoire I3S et enseignant à l'université de Nice – Sophia-Antipolis, Mr Collet nous a fait profiter de ses connaissances dans les technologies que notre projet se devait d'utiliser, nous mettant ainsi sur la bonne voie à suivre dés le début. Il nous a également guider et conseiller tout au long de ce TER.

II . Cahier des charges synthétiques

II.1 Objectifs

Fractal TV est un projet visant à développer un prototype client/serveur d'architecture 1-N, proposant des vidéos à la demande par deux modes de transmission :

- Téléchargements
- Streaming (protocole RTP)

Les fonctionnalités prévues initialement devaient permettre de :

- choisir le type de diffusion
- paramétrage et lecture des vidéos
- savoir lire les vidéos en fonction des codecs installés sur la machine
- négocier la qualité de la vidéo en fonction de la bande passante disponible
- tester la charge du serveur

II.2 Contraintes

Le sujet nous imposait d'utiliser le langage Java, l'utilisation de la plate-forme de composants Fractal et notamment son implémentation en Java, la Java Média Framework, et enfin la programmation répartie par l'intermédiaire de fractal RMI.

Il nous a fallu assurer également une certaine portabilité du prototype qui doit pouvoir tourner sur Windows, Linux et Windows Mobile 2003.

Nous devions permettre la réutilisation du code grâce à la programmation par composants.

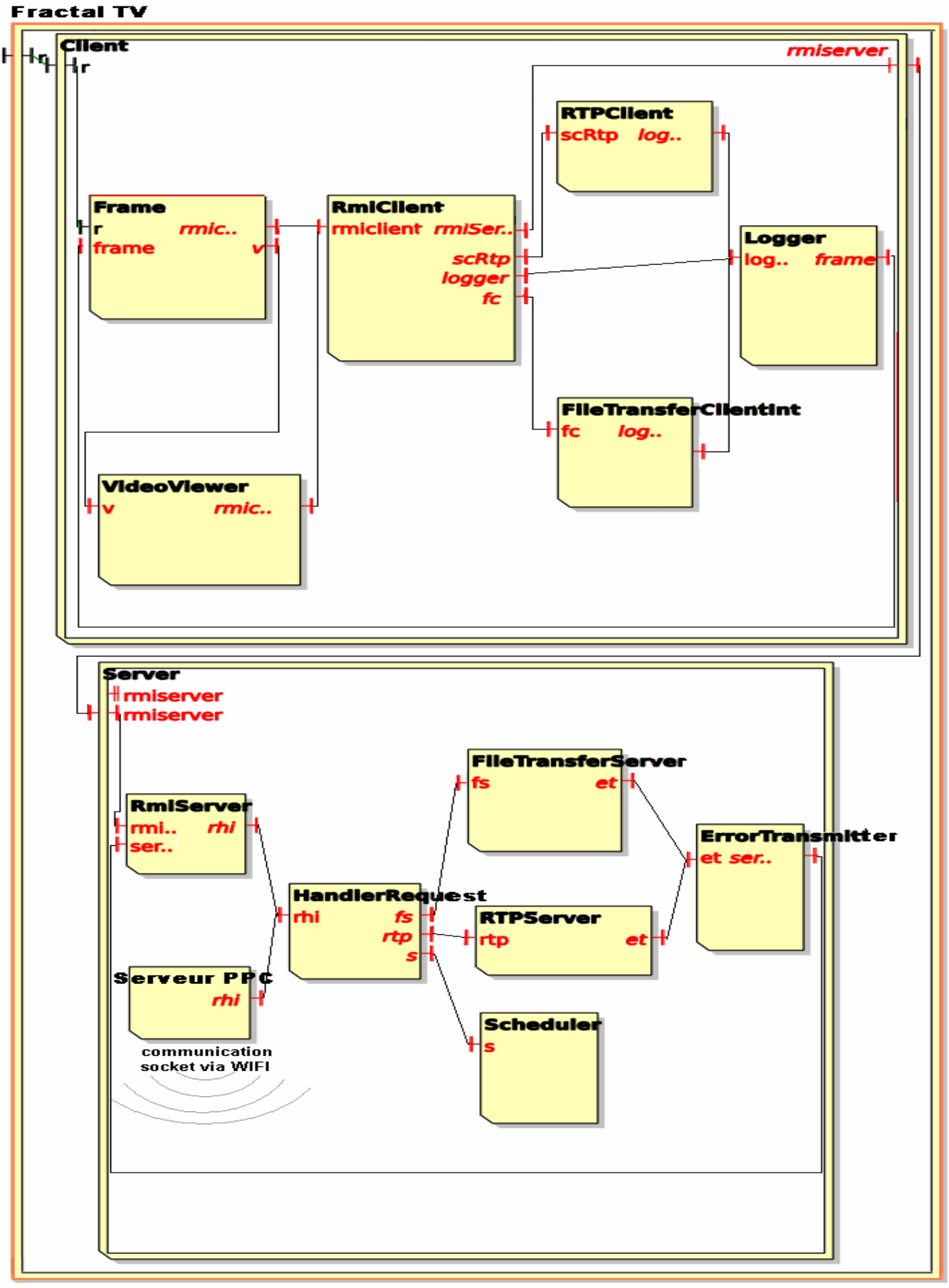
II.3 Gestion des risques

Le risque le plus important identifié lors de la mise en place du cahier des charges était le Pocket PC, le déploiement de toutes les technologies requises sur ce type d'architecture est compliqué et nous était totalement inconnu, ce risque aurait pu nous amener à abandonner cette partie du projet.

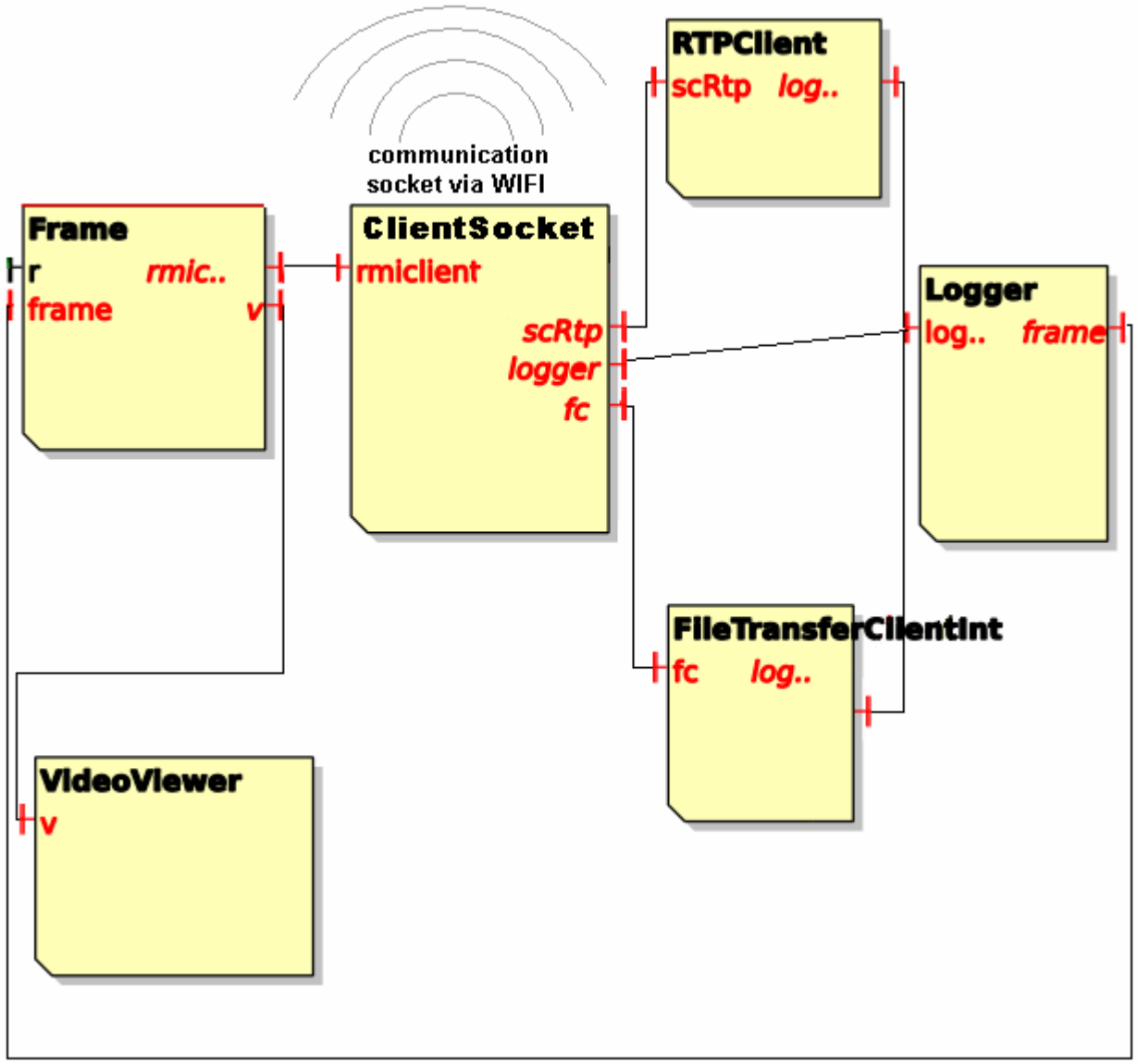
Un autre risque était les performances que l'on pourrait avoir sur ce type de machine, un projet de l'ESSI visait à établir une version performante du décodage vidéo avec la JMF devait nous permettre de remédier à ce problème.

III . Architecture finale de notre prototype

III.1 . Architecture Client-Serveur PC



III.2 . Architecture Client Pocket PC



IV . Planning

IV.1 Calendrier prévisionnel

Après avoir étudier les risques, les objectifs à remplir et après une petite expérimentation de toutes les technologies requises, nous nous sommes mis d'accord sur une approche incrémentale de la phase de développement et par conséquent nous avions décider de découper le planning comme tel :

Nom	***Nicolas Yohann***	***Barbier Thomas***	***Castillejos Nicolas***	***Salageanu Emil***	***Sauvan Bastien***
Semaine 1	Fractalisation et modifications du serveur RTP	Fractalisation Serveur Téléchargement et assemblage des différentes parties	Fractalisation du client RTP	Fractalisation de la partie RMI (en utilisant toujours Java RMI) et assemblage	Fractalisation de la partie RMI (en utilisant toujours Java RMI) et assemblage
Semaine 2	Adaptation en Fractal RMI	Adaptation en Fractal RMI	PocketPC et installation de la J9	Adaptation en Fractal RMI	Adaptation en Fractal RMI
Semaine 3	Fractalisation en composant plus détaillée du serveur RTP + Déploiement par ADL	Fractalisation en composant plus détaillée RTP et/ou téléchargement. (éventuellement sur la J9)	PocketPC et installation de la J9	Fractalisation en composant plus détaillée de la partie RMI	Fractalisation en composant plus détaillée de la partie RMI
Semaine 4	Assemblage, déploiement et test	Assemblage, déploiement et test	PocketPC et installation de la J9 (et/ou déploiement)	Assemblage, déploiement et test	Assemblage, déploiement et test
Semaine 5	Semaine de réserve où on place les efforts sur les points faibles de l'application.	Semaine de réserve où on place les efforts sur les points faibles de l'application.	Semaine de réserve où on place les efforts sur les points faibles de l'application.	Semaine de réserve où on place les efforts sur les points faibles de l'application.	Semaine de réserve où on place les efforts sur les points faibles de l'application.
Semaine 6	Finitions, rapport, repérage de bugs de dernière minute.	Finitions, rapport, repérage de bugs de dernière minute.	Finitions, rapport, repérage de bugs de dernière minute.	Finitions, rapport, repérage de bugs de dernière minute.	Finitions, rapport, repérage de bugs de dernière minute.

IV.2 Planning effectif

Voici le planning effectif que nous avons suivi lors de la phase de développement :

Nom	***Nicolas Yohann***	***Barbier Thomas***	***Castillejos Nicolas***	***Salageanu Emil***	***Sauvan Bastien***
Semaine 1	Récupération du code effectué lors de la phase de conception et transformation en composants Fractal détaillés du Serveur RTP	Transformation du Serveur de Téléchargements en composants Fractal	Fractalisation du client RTP en composant de haut - niveaux et installation de la J9 sur Pocket PC	Fractalisation de la partie RMI (en utilisant toujours Java RMI) et assemblage d'un premier prototype	Fractalisation de la partie RMI (en utilisant toujours Java RMI) et assemblage d'un premier prototype
Semaine 2	Initialisation au déploiement en ADL, apprentissage de l'outil fractalGUI et déploiement ADL du Serveur RTP	Conception d'une interface graphique minimaliste et initiation à l'ADL	Premiers essais de lecture RTP et première application fractal sur Pocket PC	1ere version FractalRMI basé sur l'exemple fourni avec FractalALL (non définitive)	Assemblage d'une deuxième version du prototype incluant l'IG. Recherches sur Fractal-RMI
Semaine 3	Déploiement par ADL de l'ensemble du prototype	Gestion dynamique des ports pour téléchargements et Streaming	Codage d'un client RTP sur PocketPC + divers tests de compatibilité JMF	Tentative d'installation de plugins JMF pour supporter plus de codecs (Codec IBM)	Assemblage avec le groupe du nouveau prototype. Travail sur le déploiement par ADL sans succès
Semaine 4	Passage en Fractal RMI version 2 + modification de l'architecture	tests et correction de bug mineur (registry et synchronized). Intégration de modifications sur le nouveau prototype)	Test de prototype sur Pocket PC et tests de compatibilité Fractal-RMI	application de test indépendante de l'application fractal, qui essaye de trouver les plugins pour un flux d'entré	Support de codecs pour notre application (ffmpeg de SourceForge). Recherches sur la capture de webcam
Semaine 5	travail sur la robustesse du prototype + téléchargement dynamique de code	Adoption de la solution de replis pour le pocket PC : création d'un composant Fractal "serveur" qui communique par socket. Modifications dans l'architecture du serveur, ADL des nouveaux composants	Adoption de le solution de replis pour le pocket PC : création d'un composant Fractal "client" qui communique par socket au serveur	Etude du JMF wrapper. Des essais sur les formats vidéo	Tests du prototype + Support webcam pour serveur. Assemblage du prototype final
Semaine 6	Finitions, rapport, repérage de bugs de dernière minute.	Finitions, rapport, repérage de bugs de dernière minute.	Finitions, rapport, repérage de bugs de dernière minute.	Finitions, rapport, repérage de bugs de dernière minute.	Finitions, rapport, repérage de bugs de dernière minute.

IV.3 Analyse

D'une manière générale, le planning prévisionnel a subi de nombreux changements dus à une mauvaise appréciation de départ et des points du projet qui ont changés de priorités au cours de la phase développement.

Tout d'abord, le déploiement du prototype par ADL a été placé par notre encadrant dans une priorité plus forte qu'elle ne l'était au départ, donc dès la deuxième semaine, un intervenant s'est vu affecter cette tâche.

Ensuite, la passage à Fractal RMI s'est terminé plus tard que prévu dû à une mauvaise conception de départ lors de la deuxième semaine et à dû être repris lors de la quatrième semaine.

A l'inverse, l'installation de la J9 sur Pocket PC s'est avérée plus rapide que prévue et a permis a Nicolas d'attaquer plus rapidement les premiers tests. Cela a permis de déterminer assez tôt l'impossibilité de faire un code unique entre un client « normal » et un client se connectant via le Pocket PC dû, entre autres, à l'absence de parseurs XML utilisés par la plate-forme Fractal sur la J9.

Lors de la troisième semaine, nous avions prévu de créer des composants Fractal plus détaillés. Cette étape n'a pas été nécessaire car elle a été effectuée lors de la première semaine. Les composants Fractal que nous avons créés ont gardé leur niveau de détails. Nous avons donc profiter de cette semaine pour se concentrer sur les points délicats de l'application (déploiement par ADL, Gestion des ports, assemblage).

Nous avons également modifié le nombre d'intervenants qui se sont occupés de la phase de tests lors de la quatrième semaine, en effet, d'autres points plus importants étaient encore à gérer et nous ne pouvions pas nous permettre de les négliger. (Fractal RMI et codecs JMF)

La semaine de réserve a été utilisée pour permettre à Nicolas et à Thomas d'adopter la solution de replis pour le Pocket PC, après l'échec des tests de Fractal RMI sur ce type d'architecture. Pour les autres, elle a permis d'étoffer les fonctionnalités du serveur avec le support de la webcam et du téléchargement dynamique de classes mais également de travailler sur la robustesse du serveur et le confort du client.

La dernière semaine a été respectée vis-à-vis du planning prévisionnel et a donc servi à faire la rédaction du rapport, à détecter des bugs, et à faire des tests.

V . Découpage en tâche et description du travail réalisé

Dans ce qui suit, nous allons vous présenter tâche par tâche, la description du travail qui a été réalisé pendant la conception de ce prototype. Pour mieux vous situer dans la chronologie de la période de développement, vous pouvez vous reporter à la section précédente avec le planning effectif.

Voici donc les différentes tâches réalisées par notre équipe :

<table>
<tr><th colspan="4">"Fractalisation" du premier prototype</th></tr>
<tr><td>Durée</td><td colspan="3">1 semaine</td></tr>
<tr><td>Nombre de personnes impliquées</td><td colspan="3">5</td></tr>
<tr><td>Taux d'implication des personnes</td><td></td><td>durée</td><td>pourcentage de son temps</td></tr>
<tr><td></td><td>Barbier Thomas</td><td>4 jours</td><td>80%</td></tr>
<tr><td></td><td>Castillejos Nicolas</td><td>1 jour</td><td>100%</td></tr>
<tr><td></td><td>Nicolas Yohann</td><td>5 jours</td><td>100%</td></tr>
<tr><td></td><td>Salageanu Emil</td><td>5 jours</td><td>100%</td></tr>
<tr><td></td><td>Sauvan Bastien</td><td>5 jours</td><td>100%</td></tr>
<tr><td>Description exacte de la tâche</td><td colspan="3">La "Fractalisation" a consisté à transformer et décomposer notre premier prototype en assemblage de composants Fractal. Nous avions déjà bien décomposé notre premier prototype en différents Objets Java et le passage en composant Fractal a donc été assez simple. Chaque membre de l'équipe a donc transformé en composants fractal la partie qu'il avait écrite lors de la première phase du TER.</td></tr>
<tr><td>Choix et problèmes techniques rencontrés</td><td colspan="3">Pas de problèmes techniques de ce côté là grâce à une initiation à l'API Fractal pendant la période de pré développement, cependant il nous fallait mettre en place une certaine cohérence entre les différents composants qui permettrait par la suite des modifications minimes dans le cas ou on voudrait modifier l'implémentation d'un composant. Par exemple, on pourrait modifier dans le futur l'implémentation du transmetteur de flux RTP pour le transforme en transmetteur « DataSink », c'est-à-dire que la vidéo s'enregistre en local en même temps que la lecture en streaming.</td></tr>
</table>

Déploiement par ADL			
Durée	2 semaines		
Nombre de personnes impliquées	2		
Taux d'implication des personnes		**Durée**	**pourcentage de son temps**
	Barbier Thomas	2 semaines	20%
	Nicolas Yohann	2 semaines	100%
Description exacte de la tâche	Il s'agissait dans un premier temps de se familiariser avec l'application Fractal-GUI fournie par l'API Fractal qui permet de générer tous les fichiers .xml nécessaires au déploiement ADL à partir du dessin graphique de l'assemblage des composants. Et dans un deuxième temps d'arriver à s'en servir pour démarrer notre application. Pour cela, il nous faut créer une « Usine » de création de composants grâce à la classe FactoryFactory et à la méthode getFactory(). Une fois « l'usine » mis en place, la méthode newComponent() permet de récupérer les fichiers ADL et toutes les références nécessaires pour créer le template.		
Choix et problèmes techniques rencontrés	Ce TER est le premier déploiement d'application par ADL que nous expérimentons, c'est un domaine qui nous est donc totalement inconnu. Les problèmes techniques seraient plutôt liés aux versions des bibliothèques fractal utilisées, une fois les bonnes versions trouvées et l'outil FractalGUI maîtrisé, nous n'avons plus rencontré de réelles difficultés du point de vue technique, je pense que la réelle épreuve a été l'apprentissage par des tutoriaux et tests divers pour réussir à déployer notre application.		
Tâche non complétée ou plus longue que prévue	Les différents composants du prototype sont déployés par cette méthode, cependant, étant donné que le client récupère une référence distante du serveur, La liaison du client au serveur s'effectue dans la méthode main du client et non dans l'ADL, en effet, il ne nous a été impossible de permettre à l'application de faire cette opération, il faut avant tout créer le composant qui « englobe » toute l'architecture du prototype, pour pouvoir y ajouter le serveur.		

Passage du prototype en Fractal-RMI			
Durée	2 semaines		
Nombre de personnes impliquées	3		
Taux d'implication des personnes		**durée**	**pourcentage de son temps**
	Nicolas Yohann	1 semaine	100%
	Salageanu Emil	1 semaine	100%
	Sauvan Bastien	5 jours	70%
Description exacte de la tâche	Cette opération consistait à passer le tout premier prototype qui marchait à l'époque avec du RMI (Java) en FractalRMI, qui est une technologie fournie par l'API Fractal qui permet d'assembler des composants Fractal même si ceux-ci sont distants. Il fallait ensuite faire cohabiter le déploiement par ADL et FractalRMI.		
Choix et problèmes techniques rencontrés	Les problèmes techniques rencontrés sont plus au niveau de l'architecture, en effet Emil avait mis en place un prototype où le client et le serveur communiquaient par fractal RMI, le problème était que le client créait, a distance, le serveur, donc incompatible avec notre objectif. Yohann a donc modifié le prototype, dorénavant, le serveur est donc du côté serveur, le template Serveur est enregistré auprès du registry, cette instance serveur est récupérée du coté client dans un composant de haut niveau et lié au composant "client".		

Installation de la J9 sur Pocket PC			
Durée	3 jours		
Nombre de personnes impliquées	1		
Taux d'implication des personnes		**Durée**	**pourcentage de son temps**
	Castillejos Nicolas	3 jours	100%
Description exacte de la tâche	Il fallait ici allé chercher les fichiers nécessaires au bon fonctionnement de la J9 contenus dans le logiciel *"IBM Websphere Studio Device Developer"* et les copier sur le Pocket PC. Il fallait ensuite s'assurer de ce bon fonctionnement en concevant 2 programmes "Hello World" : 1 avec un simple System.out.println et 1 dans une fenêtre AWT.		
Choix et problèmes techniques rencontrés	Pas de problème rencontré à ce niveau.		

Tests de lecture vidéo et tests de l'API Fractal sur Pocket PC			
Durée	2 semaines		
Nombre de personnes impliquées	1		
Taux d'implication des personnes		**Durée**	**pourcentage de son temps**
	Castillejos Nicolas	2 semaines	100%
Description exacte de la tâche	Il s'agissait ici d'arriver à lire un flux RTP envoyé depuis un PC sur le pocket PC mais aussi de tester l'API Fractal à l'aide d'un exemple simple.		
Choix et problèmes techniques rencontrés	On savait déjà que la JMF (Java Media Framework), l'API java que nous avions déjà expérimentés sur PC lisait les vidéos à une vitesse extrêmement lente sur Pocket PC grâce à une démo de notre encadrant. La première étape a donc été de chercher une alternative à la JMF. La seule trouvée était d'arriver à faire lire les vidéos par le "Windows Media Player" déjà présent sur le PC à l'aide de JNI. Mais on s'est rendu compte que le lecteur Windows Media Player ne lisait pas directement du RTP et qu'il fallait en venir à de plus amples modifications de notre application notamment du côté serveur pour arriver à nos fins. On a donc laissé cette idée en suspens pour y revenir plus tard si le temps nous y laissait l'occasion. On a alors décidé de commencer par une solution plus raisonnable : utiliser la JMF mais en faisant passer de la vidéo encodé avec le codec H263 qui utilise peu de CPU lors du décodage. Il a fallut donc concevoir un mini client RTP pour Pocket PC. Beaucoup de problèmes ont été rencontrés avec l'utilisation de la JMF sur Pocket PC car celle-ci n'est pas du tout prévue pour fonctionner avec la J9 : des exceptions apparaissant à l'initiation de la lecture vidéo ou audio (ClassNotFoundEsception, Unable to handle format ou Cannot Realize ...). Ces problèmes sont principalement dus à l'absence ou présence de classes dans le .jar de la JMF. On a réussi à résoudre certains de ces problèmes en cherchant sur des forums qui indiquait précisément quelle classe il fallait ajouter ou enlever pour que cela fonctionne (ex : pour faire fonctionner la vidéo il est nécessaire d'enlever la class Java2DRenderer du .jar de la JMF). En ce qui concerne le test de l'exemple simple en Fractal, il n'y a eu aucun problème.		
Tâche non complétée ou plus longue que prévue	Contrairement à la lecture vidéo, la lecture audio n'a pas pu être résolue, même en testant des solutions données par quelques personnes sur les forums de discussions concernant la JMF. Les seules personnes qui sont arrivées à lire de l'audio sur Pocket PC ne travaillait pas avec la J9 (Creme VM ou Jeode VM) et n'arrivent qu'a lire des .wav en local. Pour information un projet de 2 élèves de l'ESSI en rapport avec la JMF a conclu que ce n'était pas possible de lire du son avec la JMF sur la J9, et de nombreuses personnes ont ce même problème sur le peu de forum existant concernant l'utilisation de la JMF sur Pocket PC.		

Etude et utilisation des codecs vidéo par la JMF			
Durée	2 semaines		
Nombre de personnes impliquées	2		
Taux d'implication des personnes		**Durée**	**pourcentage de son temps**
	Salageanu Emil	2 semaines	50%
	Sauvan Bastien	2 semaines	100%
Description exacte de la tâche	La JMF possédant un nombre très limité de format vidéos compatibles, Emil et Bastien se sont mis à étudier la gestion des codecs par la JMF dans le but de permettre à notre application d'être compatible avec une multitude de formats.		
Choix et problèmes techniques rencontrés	Les recherches sur des documentations et exemples de gestion de codecs par la JMF ayant été peu fructueuses, cette partie s'est avérée délicate. Dans un premier temps, nous avons essayé d'installer et d'utiliser la librairie de codec d'IBM. L'installation fut rapidement réalisée mais nous n'avons pas pu trouver, lors des tests, de format supplémentaire que le prototype ou même le lecteur de la JMF, JMStudio, ne pouvaient lire auparavant. Parallèlement, nous avons testé la librairie ffmepg de SourceForge. La aussi, son installation fut vite mise en place. Les test se sont avérés toutefois plus concluant, nous avons pu de suite lire avec le lecteur JMStudio certains fichiers avi et mpg supplémentaires (Le lecteur ne pouvant lire au départ des certains fichiers mpg et certains fichiers quicktime). Puis après quelques manipulations de librairies pour intégrer ces codecs dans le prototype, nous avons pu lire ces mêmes fichiers avi en lecture locale, ainsi que des fichiers mpg en streaming et en local.		
Tâche non complétée ou plus longue que prévue	La lecture en streaming de fichiers avi n'a pas pu être réalisée car nous n'avons pas trouvé de librairies de codecs adéquate ni même d'informations concrètes à ce sujet. De plus, la lecture en streaming ou même en local de certains fichiers mpg ou avi ne marchent pas certainement pour des raisons d'encodages différents de ceux supportés par les codecs installés.		

Adaptation du prototype sur Pocket PC			
Durée	2 semaines		
Nb de personnes	1		
Taux d'implication des personnes		**Durée**	**pourcentage de son temps**
	Castillejos Nicolas	2 semaines	100%
Description exacte de la tâche	Il fallait ici arriver à faire fonctionner correctement le dernier prototype en date fonctionnel sur le Pocket PC (uniquement le client)		
Choix et problèmes techniques rencontrés	Le principal problème rencontré est l'incompatibilité entre la partie "Fractal-RMI" de l'API Fractal et la J9 VM dIBM. En effet Fractal-RMI utilise du parsing XML pour fonctionner et nécessite des classes de bases concernant le XML qui ne sont apparues qu'à partir de Java 1.4 et ne figurent donc pas dans la J9 (qui est basée sur Java 1.3). On ne s'est pas arrêté là, on a alors essayé de se fabriquer un .jar des classes manquantes en allant les récupérer depuis java 1.4 et d'inclure ce .jar dans le classpath. Pour information les classes manquantes appartiennent aux packages : org.xml.sax, org.apache.crimson, javax.xml.transform, javax.xml.parsers. le rajout de ces classes a donc permis de corriger les NoClassDeFoundError apparaissaient au lancement. Une autre erreur est apparue ensuite : "*Cannot find or instantiate the 'org.objectweb.fractal.julia.loader.DynamicLoader' class*". Ceci était dû à une mauvaise version du .jar julia-runtime.jar. Le script ant qui compile ce .jar crée en fait 2 versions de ce .jar : 1 version pc julia-runtime.jar et une version j2me julia-runtime-j2me.jar. c'est cette dernière qu'il faut utiliser pour contrer cette erreur. Une fois cette erreur corrigée, une autre est apparue : *java.security.AccesControlException: Access denied (java.lang.RuntimePermission createClassLoader)* qui viendrait apparemment du fait que la J9 ne tient pas compte du fichier java.policy qu'on lui spécifie. Plusieurs tests ont été essayé pour tenter de résoudre le problème mais celui-ci est resté sans solution.(on a essayé de le définir dans la commande de lancement, ou par un System.setProperty, ou de placer le fichier java.policy dans le répertoire \J9\lib\security comme un site l' indiquait de le faire mais sans résultat). On a ensuite essayé de ne pas définir de Security Manager (même si ça devait planté plus loin car Fractal-RMI nécessite un Security Manager pour fonctionner tout comme RMI) dans le but de voir si c'était la seul erreur qu'il restait à corriger... Mais il est alors apparu une erreur de parsing XML comme quoi la J9 n'arrivait pas à parser un fichier .adl contenu dans un .jar de Fractal, elle n'arrivait pas a lire du format UTF8 alors que les fichiers .adl contenu dans le .jar contenait du format ISO... On a essayé en parallèle la même chose avec le jdk 1.3 sur PC et là pas de problème de format de fichier .adl ni de fichier java.policy pas pris en compte, mais un problème de sécurité au moment de la désérialisation par le stub (java.lang.SecurityException: Prohibited package name: java.lang [java]**)**. Arrivé ici, une semaine s'est écoulée et la fin du TER s'approchant à grand pas nous avons décidé d'adopter une solution de replis afin de pouvoir fournir un client pour Pocket PC fonctionnel. Cette solution de replis est donc d'utiliser la communication par socket au lieu de la commnunication Fractal-RMI. On garde toujours des composants Fractal mais le déploiement se fait sans fichier XML, en dur dans un main par des bindfc. Cette solution de replis n'a pas posé de problème et nous a permis d'obtenir un client Fonctionnel de notre application pour Pocket PC.		
Tâche non complétée ou plus longue que prévue	Cette tâche a été beaucoup plus longue que prévue, car nous avions testé auparavant un exemple simple de Fractal et celui-ci marchait parfaitement. Nous nous étions donc dit que Fractal n'allait pas nous poser se problème. Mais c'est en testant le prototype qui utilise beaucoup plus de fonctionnalité de Fractal que l'exemple testé qu'on s'est rendu compte qu'il y'avait une incompatibilité certaine entre la partie Fractal-RMI et la J9.		

Améliorations et finalisation du prototype			
Durée	1 semaine		
Nb de personnes	4		
Taux d'implication des personnes		**Durée**	**pourcentage de son temps**
	Barbier Thomas	2 semaines	100 %
	Castillejos Nicolas	3 jours	100%
	Nicolas Yohann	3 jours	50 %
	Sauvan Bastien	4 jours	50%
Description exacte de la tâche	Cette tâche consistait à : - Permettre à l'utilisateur de comprendre un minimum ce qui se passe quand une vidéo ne peut pas être envoyé par Streaming ou lors d'un plantage du serveur par l'utilisation d'un nouveau composant Logger qui permet l'affichage d'une console d'exécution affichant tout ce qui se passe lors des interactions client/serveur. - Adapter le serveur pour qu'il puisse envoyer des vidéos au format compatible Pocket PC (H263 à 4images/sec bitrate minimal) - Concevoir et connecter le nouveau composant ServerSocket du côté serveur qui permet de communiquer avec les clients Pocket PC - Améliorer l'interface graphique - L'ajout du support Webcam côté serveur - L'attribution des ports pour les communications RTP		
Choix et problèmes techniques rencontrés	- Le logger : Pour pouvoir faire des appels de méthodes sur le client, le serveur a besoin de garder des références de tous ses clients, le serveur doit donc "connaître" la classe Client.Yohann a donc mis en place des minis serveurs http du coté du client et du serveur permettant a l'un comme a l'autre de télécharger les classes dynamiquement,cette méthode marche mais il faut que le réseau soit configurer correctement : la mise en place du codebase fait à une méthode getHostName() qui dépend de l'OS qui tourne derrière, sous linux par exemple, elle dépend du fichier /etc/hosts .. - Le format compatible Pocket PC : il s'est avéré que le redimensionnent de la vidéo lors de l'encodage aux formats compatibles H263 (SQCIF (128x96), QCIF (176x144) et CIF (352x288)) n'a pas fonctionné sous Linux à la fac, mais fonctionne très bien si le serveur tourne sous Windows.		

Les Tests			
Durée	1 semaine		
Nb de personnes	5		
Taux d'implication des personnes		**Durée**	**pourcentage de son temps**
	Barbier Thomas	3 jours	50%
	Castillejos Nicolas	3 jours	50 %
	Nicolas Yohann	3 jours	50 %
	Salageanu Emil	3 jours	50 %
	Sauvan Bastien	3 jours	50 %
Description exacte de la tâche	Nous devons ici tester la charge de l'application : plusieurs client qui se connecte au même serveur et demande de l'envoie streaming de vidéo et du téléchargement de vidéo en même temps. Nous devons aussi écrire en détail ce qu'il faut faire pour pouvoir reproduire ces tests afin de vérifier le bon fonctionnement de l'application.		
Choix et problèmes techniques rencontrés	Nous nous sommes réunis à la fac pour pouvoir tester les capacités du serveur, mais également le bon fonctionnement des clients, tout d'abord, nous avons lancé un serveur, puis on y a connecté 3 clients, chacun de ces clients a lancé environ 5-6 demandes de vidéo par Streaming et quelques demandes par téléchargement, le test a été concluant même si la qualité de la vidéo diffusé dépendra beaucoup de la bande passante du serveur. Nous avons également effectué des tests avec des vidéos qui ne peuvent pas être transmises par RTP, afin de s'assurer que le client ne reste pas bloqué en attente de la vidéo et que l'utilisateur comprenne ce qui se passe.		

Conception du rapport et des fournitures			
Durée	1 semaine		
Nb de personnes	5		
Taux d'implication des personnes		**Durée**	**pourcentage de son temps**
	Barbier Thomas	1 semaine	100%
	Castillejos Nicolas	1 semaine	80%
	Nicolas Yohann	1 semaine	100%
	Salageanu Emil	2 jours	30%
	Sauvan Bastien	1 semaine	75 %
Description exacte de la tâche	Ceci concerne la rédaction de ce rapport ainsi que les différentes fournitures et documents rédigés qui formeront le "Livrable" de ce TER		
Choix et problèmes techniques rencontrés	Des scripts ant ont été mis en place pour assurer une certaine facilité d'utilisation, les bibliothèques utilisées par le projet dépendent du système d'exploitation qui est derrière.		

VI . Bilan

V.1 Fonctionnalités

Après six semaines de développement, nous avons un prototype qui offre les fonctionnalités suivantes :

Nous disposons d'un serveur proposant des vidéos en téléchargement ou en streaming. Ce serveur utilise FractalRMI et fonctionne sous linux et windows. En ce qui concerne le client, la version PC (Windows et Linux) utilise FractalRMI tandis que la version Pocket PC se connecte au serveur par les Socket java de base. Le serveur dispose donc d'un composant supplémentaire permettant la communication Pocket PC -> Serveur. Ainsi, nous avons pu atteindre nos objectifs principaux, à savoir un prototype minimal permettant deux modes d'utilisations clients (streaming et téléchargement).

Notre application est bien une architecture 1-N et le même serveur peut traiter les différentes requêtes provenant aussi bien d'un PC que d'un Pocket PC. L'extensibilité de notre application est assurée grâce à la programmation par composant basée sur la plateforme Fractal et à notre architecture. Elle permet de rajouter des composants frontaux au niveau du serveur, afin de pouvoir gérer de nouveaux protocoles communication Client -> Serveur , il suffit juste aux nouveaux composants frontaux d'être branché sur le RequestHandler qui fait office de composant intermédiaire déléguant les tâches aux composants RTP et téléchargement. Des tests de charge sont possibles grâce aux scripts ant fournis avec l'application.

La portabilité du code source a pu être réalisée jusqu'à un certain point. Ainsi le même client peut fonctionner sur Linux ou Windows mais pas sur PocketPc. Les problèmes que nous avons rencontré et qui sont détaillés précédemment, nous ont contraint à réaliser un autre client pour le Pocket PC. Ce nouveau client peut également marcher sur les 2 systèmes d'exploitations des PC mais son utilisation est limitée au Pocket PC afin de respecter les contraintes initiales de notre TER, c'est à dire l'utilisation totale de la plateforme Fractal.

V.2 Contraintes

Une des contraintes imposée par le cahier des charges était de pouvoir faire tourner l'application Fractal-TV sur des Pocket PC dotés de 32-64 Mo de ram voir 128 Mo pour les plus récents. Après des mesures effectuées sur le Pocket PC fourni par la faculté, seulement 15,32 Mo sons nécessaires pour pouvoir installer et faire tourner le prototype. Cela comprends les fichiers de la J9, les fichiers du prototype, et la mémoire utilisée lors d'une lecture vidéo.

Voici à titre indicatif quelques tests de charge de notre application qui peuvent faire part des contraintes techniques qu'ont les machines sur lesquels tournent les clients et le serveur de notre prototype :

Téléchargements :

50 téléchargements en même temps depuis le même client	
Taille du fichier envoyé	12,7 Mo
Temps total de l'opération	1 min 40 sec
Connexion utilisée pour le test	réseau local : 100 MB/s
Caractéristiques du client	Barton 3000+, 1 GO Ram, Windows XP
Caractéristiques du serveur	Duron 800 384 Mo RAM, Windows XP
Ram utilisée sur le serveur par l'application	51 MO
Ram utilisée sur le client par notre application	36 MO

100 téléchargements en même temps depuis le même client	
Taille du fichier envoyé	2,1 Mo
Temps total de l'opération	34 sec
Connexion utilisée pour le test	Exécution en local (client et serveur sur la même machine)
Caractéristiques du client	AMD Sempron(tm) 3000, 256 Mo RAM, Linux
Caractéristiques du serveur	AMD Sempron(tm) 3000, 256 Mo RAM, Linux
Ram utilisée sur le serveur par l'application	34.8 Mo
Ram utilisée sur le client par notre application	25 Mo

1000 téléchargements en même temps depuis le même client	
Taille du fichier envoyé	2,1 Mo
Temps total de l'opération	2 minutes 43 secondes
Connexion utilisée pour le test	Exécution en local (client et serveur sur la même machine)
Caractéristiques du client	AMD Sempron(tm) 3000, 256 Mo RAM, Linux
Caractéristiques du serveur	AMD Sempron(tm) 3000, 256 Mo RAM, Linux
Ram utilisée sur le serveur par l'application	51.2 Mo
Ram utilisée sur le client par notre application	38,4 Mo

Streaming :

En ce qui concerne le Streaming, ne pouvant pas lancer énormément de fenêtres de lecture vidéo sur le même client (à cause du décodage vidéo qui prend beaucoup plus de ressources CPU que le téléchargement par socket par exemple), nous avons fait de simples tests sur 4 machines différentes (Intel 2.5 Ghz, 512 Mo Ram, Linux) en demandant plusieurs envoie de films sur chacune des machines, et pour une 12 aines de vidéos nous n'avons pas constaté de ralentissement visible de framerate.

VII . Tâches restant à accomplir et perspectives.

La découverte à la dernière semaine du bug de la *JMF Linux performance Pack* empêchant le redimensionnent de la vidéo, nécessaire au PocketPc, nous oblige à avoir un serveur en Windows pour permettre la diffusion en streaming vers des clients PocketPC. Ce bug est dû à la JMF qui n'offre pas l'ensemble des services affirmés. Le manque de temps nous a empêché de trouver une solution, mais c'est un problème qui serait important à régler dans le futur.

La négociation de la qualité de la vidéo en fonction de la charge du serveur et/ou la demande du client n'a pas été implémenté par manque de temps. Les problèmes d'architectures nous ont obligé a nous recentrer sur des parties plus fondamentales du prototype. L'ajout de cette fonctionnalité permettrait un meilleur confort de visualisation au client, ainsi qu'une meilleur capacité (en diminuant proportionnellement la qualité de la vidéo en fonction du nombre de connexions).

Une gestion d'une panoplie de codecs différents aurait étendu la bibliothèque de vidéo que pourrait distribuer le serveur en streaming. Malheureusement, cela n'a été réalisé que pour la lecture en local. Notamment par le manque d'implémentation des codecs en java disponible et l'absence d'exemple de vidéos de format non trivial diffusé en RTP ni même de documentation n'ont pu être trouvé.

VIII . Principales difficultés rencontrées

Malgré notre estimation du coût technologique, des problèmes inattendus sont survenus en dehors des problèmes de compatibilité, entre autre, la JMF et l'API fractal ne réalisaient pas l'ensemble des fonctionnalités annoncées.

De plus, nous avons constaté une incompatibilité entre les différentes versions des librairies Fractal, ainsi il nous a fallu du temps pour trouver un ensemble de librairies cohérentes entre elles.

Dans le cadre de l'API JMF, il a été nécessaire d'utiliser les librairies spécifiques à chaque plate-forme en raison du manque important de fonctionnalités de la *cross-platform,* utilisée d'ailleurs pour le Pocket PC

Le déploiement par ADL a été difficile à assimiler, notamment à cause de son originalité et la difficulté à trouver une version de F*ractalgui* entièrement fonctionnelle.

Au niveau du Pocket PC, nous avons rencontré de nombreux problèmes de compatibilité entre la machine virtuelle (la J9) et les deux API imposées (JMF, FractalRMI).

Et pour conclure, le manque de documentation concernant le développement pour Pocket PC ne nous a pas permis de résoudre tous les problèmes que l'on a rencontrés lors de ces six semaines.

IX . Compétences acquises

Ce projet nous a permis de nous initier à la programmation par composant par la plateforme Fractal et nous a donné une approche du développement en système embarqué, on s'est rendu compte que ce dernier était très fastidieux, chaque étape étant un nouveau défi à chaque fois car il fait appel à de nouvelles fonctionnalités pas toujours compatible dans certains cas. Sur un autre point, le TER nous a également permis de faire une introduction dans le monde du multimédia par la Java Média Framework, ses inconvénients, ses avantages et les différentes recherches que nous avons dû entreprendre pour nous initier à cette API ont été très formateur.

X . Conseils pour poursuivre le développement

Par expérience nous avons remarqué l'importance d'avoir une démarche rigoureuse lors du choix des librairies de Fractal (bien faire attention aux versions, à prendre des versions cohérentes entre elles).

De même il ne faut pas sous-estimer le temps nécessaire à l'assemblage, même s'il s'agit de programmation par composant, l'assemblage et le déploiement du prototype complet pouvant s'avérer très fastidieux. Notamment à cause de la difficulté pour détecter les erreurs à l'exécution (tracer les binds des composants, tester les fonctionnalités des différentes API séparément...).

Il est important de commencer par des petits exemples qui utilisent au fur et à mesure de plus en plus de fonctionnalités des API pour bien évaluer la difficulté étape par étape.

En ce qui concerne le Pocket PC, la lenteur du décodage de la JMF cross-plateform sur Pocket PC ne permettant pas d'avoir une qualité suffisante pour pouvoir regarder une vidéo convenablement, nous conseillons donc à une équipe qui voudrait avoir de la vidéo opérationnelle sur Pocket PC d'utiliser une JMF optimisée pour Pocket PC. Il n'en n'existe pas à ce jour mais il n'est pas impossible d'en voir arriver une un jour. Vérifiez aussi sur les forums parlant de la JMF sur la J9 qu'un Tweak n'est pas été trouvé entre temps pour pouvoir lire de l'audio sous Pocket PC.

De même pour l'incompatibilité entre Fractal-RMI et la J9, nous vous conseillons de vérifier que les développeurs Fractal n'aient pas sortit entre temps une nouvelle version de Fractal entièrement compatible avec la J9. Dans ce cas il serait plus pratique de revenir à un code unique sur PC et Pocket PC.

XI . Références bibliographiques

XI.1 Liens JMF:

Site officiel de la JMF
http://java.sun.com/products/java-media/jmf/index.jsp

Javadoc de l'API JMF
http://java.sun.com/products/java-media/jmf/2.1.1/apidocs/index.html

Forum sur la JMF
http://forum.java.sun.com/forum.jsp?forum=28

Un guide sur la JMF qui date de la v2.0 mais qui donne tous les principes de fonctionnement interne de la JMF
http://java.sun.com/products/java-media/jmf/2.1.1/guide/

Sites et Forums supplémentaires utilisées pour la JMF:
http://www.laas.fr/~gauriol/enseignements/Fichiers_JMF/coursJMFPerso.pdf
http://www.javafr.com/recherche.aspx?r=jmf&tr=tout

Forums pour la gestion de la vidéo avec JMF:
http://www.javafr.com/gma/JMF%20video

Forums pour la gestion de la webcam avec la JMF:
http://www.javafr.com/gma/jmf%20webcam

Pour les codecs:
http://www.alphaworks.ibm.com/tech/mpeg-4
http://sourceforge.net/projects/jffmpeg/

Forum officiel Sun pour JMF
http://forum.java.sun.com/forum.jspa?forumID=28

Guide Sun JMF
http://java.sun.com/products/java-media/jmf/2.1.1/guide/

Forum JMF/RTP
http://www.tek-tips.com/threadminder.cfm?pid=785

XI.2 Liens Fractal

Site Officiel de Fractal:
http://fractal.objectweb.org/

Javadoc de l'API Fractal
http://fractal.objectweb.org/current/doc/javadoc/fractal/overview-summary.html

Site de téléchargement CVS de Fractal. Pour télécharger la plate forme complète et les différents modules
http://forge.objectweb.org/cvs/?group_id=22

Tutoriel Fractal
http://fractal.objectweb.org/tutorial/index.html

Tutoriel Julia
http://fractal.objectweb.org/tutorials/julia/index.html

Javadoc de l'API Julia
http://fractal.objectweb.org/current/doc/javadoc/julia/overview-summary.html

Javadoc de l'API Fractal RMI
http://fractal.objectweb.org/current/doc/javadoc/fractal-rmi/overview-summary.html

Tutoriel Fractal ADL
http://fractal.objectweb.org/tutorials/adl/index.html

Javadoc de l'API Fractal ADL
http://fractal.objectweb.org/current/doc/javadoc/fractal-adl/overview-summary.html

XI.3 Liens PocketPC et J9

LA VM9 d'IBM pour le Pocket PC. Il faut télécharger la version "trial du WebSphere Device Developer car on n'a besoin que de la VM pour Pocket PC
http://www-306.ibm.com/software/wireless/wsdd/

Forum pour l'installation de la VM9
http://groups.google.fr/groups?dq=&hl=fr&lr=&ie=UTF-8&threadm=bqs6oa%245t16%241%40news.boulder.ibm.com&prev=/groups%3Fhl%3Dfr%26lr%3D%26ie%3DISO-8859-1%26q%3D%26btnG%3DRecherche%2BGoogle%26meta%3Dgroup253Dibm.software.websphere.studio.device-developer

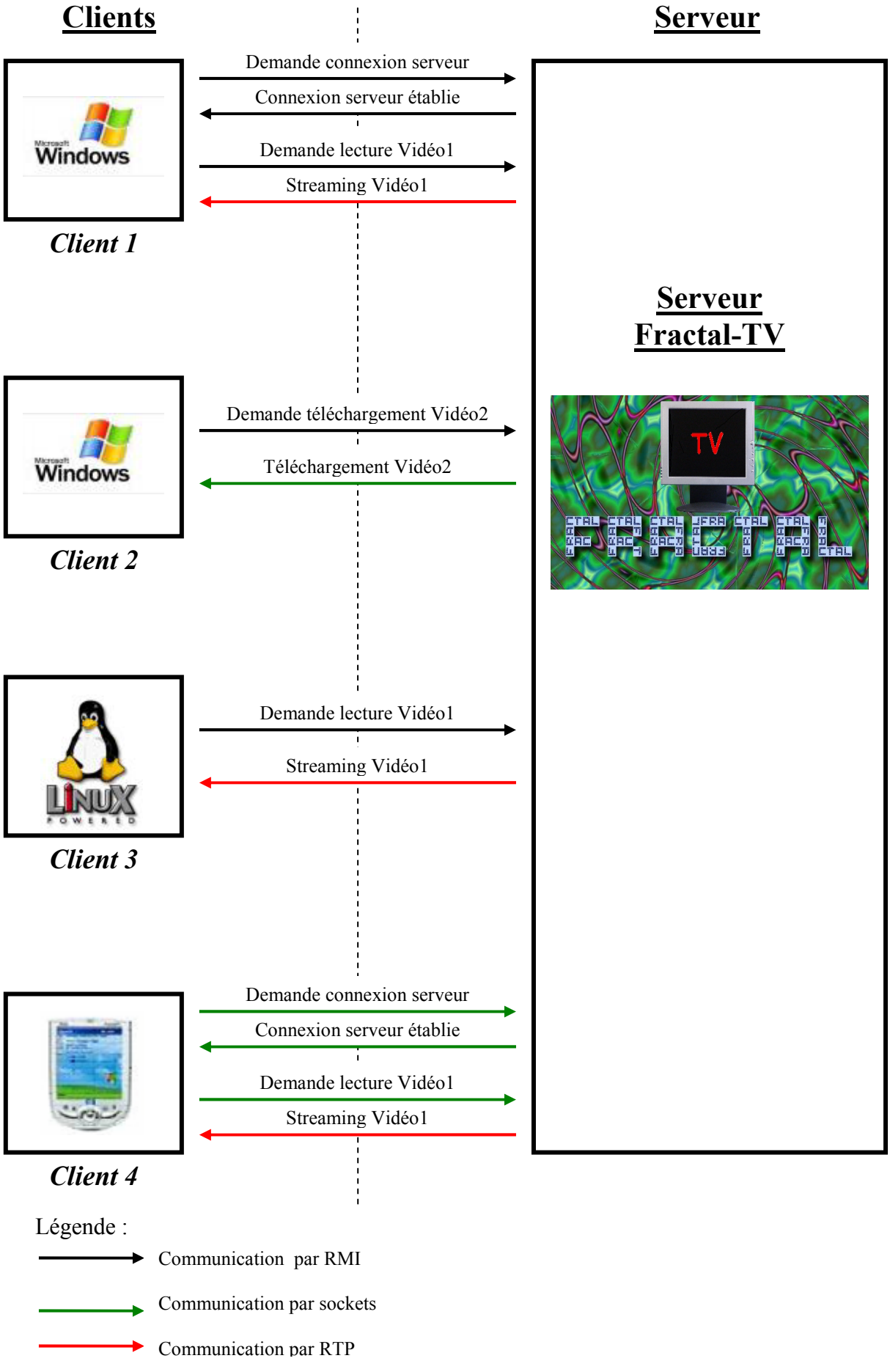
Un guide des JVM sur Pocket PC. Aller au guide d'install VM9, qui lui-même donne une référence sur un article intéressant d'IBM sur les différents modules de VM9 en fonction des versions qu'on veut installer...
http://www.berka.name/stan/jvm-ppc/java_for_pda.html

Post conseillant de supprimer la classe Java2DRenderer.class du jar de la JMF (jmf.jar) pour les PocketPcs
http://forum.java.sun.com/thread.jspa?threadID=238274&start=15

Et en documentation papier. Le rapport du projet "Java Media Framework Optimisée pour Windows Mobile sous Pocket PC" de 2 élèves de l'ESSI, Romain MAS et Steve MOREAU.

XII . Annexes

- **Schéma d'exemples de cas d'utilisation**



XIII . Autres livrables

En plus de notre application et de ses sources notre archive accompagnant ce rapport contiendra les fournitures suivantes :

- Une Javadoc
- Un manuel d'utilisation expliquant comment lancer et se servir de notre application
- Un manuel de maintenance facilitant la reprise de notre projet par d'autres développeurs
- Des tutoriaux d'installation et de paramétrage de la J9 pour la rendre compatible avec la JMF, et de programmation sur Pocket Pc
- Un environnement de test permettant de tester la charge du serveur et de tester les différentes technologies qu'utilise notre application séparément pour vérifier ce qui ne marche pas en cas de panne