Une architecture d'agents mobiles pour des réseaux de stations nomades

Andrzej Duda, Stéphane Perret
LSR-IMAG,
BP 72,
38402 Saint Martin d'Hères Cedex, France
e-mail : Andrzej.Duda@imag.fr
e-mail : Stephane.Perret@imag.fr
tél : +33 4 7682 7268
fax : +33 4 7682 7287

RESUME : Nous présentons une adaptation de l'architecture d'agents mobiles MAP à l'environnement nomade. L'approche basée sur les agents mobiles apporte plusieurs avantages par rapport aux approches traditionnelles. Elle permet le mode de travail asynchrone et la programmation d'applications par délégation. La couche d'adaptation nomade gére le collecteur des résultats comme un cache prédictif, fourni des résultats partiels ou approchés et ajoute au modèle MAP la gestion optimisée des connexions.

MOTS CLEFS : Réseaux de mobiles, Réseaux de stations nomades, Agents mobiles, Internet et applications.

1. Introduction

Traditionnellement, on considére un ordinateur comme une station sédentaire connectée de manière permanente au réseau. Cette vision est de plus en plus dépassée par l'apparition des machines nomades qui peuvent se déplacer et se connecter au réseau de manière intermittente depuis des endroits différents. La mobilité est une caractéristique essentielle des stations nomades qui peuvent utiliser une connexion sans-fil pour communiquer pendant un déplacement et un point d'accés fixe au réseau traditionnel pour se reconnecter aprés un déplacement. On peut actuellement constater une grande variété de supports de communication :

GSM à 9.6 Kbit/s),
- connexions sans-fil locales à un débit important (*wireless* LAN à 2 Mbit/s),
- connexions par modem à bas débit (V.34 à 28.8 Kbit/s),
- connexions Internet sur des distances importantes (débit variable entre Ko/s et o/s),
- connexions locales sur des réseaux locaux à un débit important (Ethernet à 10 ou 100 Mbit/s),
- connexions locales sur des réseaux locaux à haut débit (ATM de Mbit/s à Gbit/s).

Ces types de connexions variées présentent des caractéristiques différentes de débit, de latence, de taux d'erreurs, et de coûts. Il est important pour une station nomade de cacher les particularités des réseaux et d'optimiser l'usage des connexions et leur coût. Une station nomade passe par des états différents, par exemple d'un état de marche à l'état d'arrêt, ou d'un état connecté à l'état déconnecté. Le problème de changement d'états doit être traité de manière différente que dans les systèmes actuels. En effet, ces systèmes traitent une coupure de connexion ou un arrêt de site comme une exception. Dans le cas de stations nomades, la prise en compte de ce type d'événements doit devenir une règle.

Un autre problème crucial pour les stations nomades est celui des performances de communications. Une connexion réseau peut devenir une ressource critique qui limite les performances perçues par l'utilisateur. Par exemple, une application synchrone qui accède aux documents WWW à partir d'une station connectée par une liaison GSM à 9.6 Kbit/s sera limitée par ce bas débit. Même en utilisant un protocole spécialisé pour ce type de connexion, on atteint le débit utile de l'ordre de 8.7 Kbit/s [KIISKINEN96]. Seul un changement du paradigme de communication peut apporter un gain de performances significatif.

Une architecture de communication pour les stations nomades doit en premier lieu prendre en compte les besoins des applications. Parmi les applications que l'on exécute sur de telles stations, l'accés à l'information répartie est de loin la plus importante des applications réseaux. Le WWW est devenu un prototype opérationnel de l'infrastructure d'information de demain [BERNERS-LEE92]. Rechercher de l'information pertinente, découvrir des ressources dans le réseau, consulter des bases de données, exécuter des actions sur des serveurs, sont des opérations courantes. A ces applications de filtrage de l'information viendront s'ajouter des applications de filtrage des communications et de négociation des communications synchrones [MESSERSCHMITT96]. Ces nouvelles applications exploiteront l'approche de programmation par délégation. Pour ces types d'application, une station nomade doit assurer l'indépendance de la localisation, l'optimisation des ressources réseau et la gestion des connexions.

Nous proposons une approche basée sur des agents mobiles et leur exécution dans un réseau de machines. Ce paradigme de communication exploite des agents mobiles qu'une station nomade peut activer dans le réseau. Les agents mobiles effectuent des tâches pour le compte d'une application et renvoient des résultats. Ce paradigme per-

tivé des agents et récupérer les résultats ultérieurement. L'exécution des agents ainsi que les résultats de leur exécution sont persistants, c'est à dire, qu'ils ne sont pas perdus à cause d'arrêts de sites ou de coupures de connexions réseau. De cette manière une station nomade devient un terminal qui contrôle un ordinateur composé de tout le réseau - *"the network is the computer"*.

Les avantages de cette approche sont multiples. Les connexions réseau peuvent être gérées au mieux - on transfère du code exécutable sur le site de stockage de données et les actions sur ces données donnent des résultats moins volumineux que les données elles-mêmes. De plus, on peut traiter les résultats avant la transmission selon leur type, par exemple on peut les compresser ou dégrader leur qualité en diminuant la résolution d'une image. On peut aussi grouper les transmissions pour augmenter l'utilisation d'une liaison pendant toute la durée d'une connexion. Un autre aspect concerne le travail en mode asynchrone qui améliore les performances perçues par l'utilisateur. En effet, l'utilisateur peut activer des actions dans le réseau et accèder aux résultats plus tard quand ces derniers sont déjà stockés localement sur la station nomade. L'accés local peut être beaucoup plus rapide qu'à travers le réseau et les temps de réponse perçus par l'utilisateur peuvent être plus satisfaisants.

Nous avons défini et mis en œuvre une architecture d'agents mobiles appelée MAP (*Mobile Assistant Programm*ing) [PERRET96A]. Un premier prototype de MAP utilise le langage Scheme et le support du WWW [PERRET96B]. Le modèle MAP a été conçu pour des applications d'accés à l'information répartie sur des réseaux de grande envergure [PERRET96D]. MAP permet la programmation d'agents mobiles appelés assistants qui peuvent se déplacer d'un site à un autre, instancier des clones et envoyer des résultats. L'exécution des assistants est persistante et les opérations de déplacement, de clonage, et d'envoi de résultats ont une sémantique d'actions atomiques.

Dans cet article, nous analysons les problèmes soulevés par l'utilisation du modèle MAP pour des réseaux de stations nomades. Nous définissons une interface entre les applications qui sont conscientes de la nomadicité et une couche d'adaptation nomade qui représente le réseau. Cette interface permet de fournir les fonctions de base nécessaires aux applications (activation des assistants, récupération des résultats) et de gérer les problèmes liés à la nomadicité. Par exemple, une application peut spécifier quelle dégradation de qualité elle peut supporter et quelle doit être la fonction d'optimisation des coûts de connexions. Elle peut aussi spécifier comment doivent être interprétés des résultats pour fournir à l'utilisateur une approximation ou des résultats partiels.

La couche d'adaptation nomade se charge de la gestion des assistants (activation, statut, désactivation) et des connexions (contrôle des connexions, notification d'état). Pour des applications sur une station nomade, elle agit comme un intermédiaire pour l'activation des assistants et la récupération des résultats, et rend visible le collecteur des résultats modifiés selon les souhaits de l'application. L'application informe la couche sur les politiques à adopter pour la gestion de connexions.
Le reste de cet article est organisé en trois sections. D'abord, nous présentons l'archi-

de stations nomades et nous définissons la couche d'adaptation nomade pour MAP. Nous présentons l'état d'avancement de sa mise en œuvre et une application nomade d'accés à l'information répartie sur le WWW au dessus d'une liaison GSM. Nous terminons par des conclusions.

2. Architecture d'agents mobiles

L'objectif du modèle MAP (*Mobile Assistant Programming*) est de fournir des mécanismes souples pour le développement des applications accédant à l'information répartie dans des réseaux de grande envergure comme l'Internet. Le modèle prend en compte les caractéristiques de ces types de réseaux où l'information est diséminée sur un grand nombre de serveurs et où les débits sont faibles et variables dans le temps. Le support d'exécution de MAP est composé d'un réseau de nœuds interconnectés par des liens de communication. Les nœuds sont des processeurs virtuels disposant d'un dispositif de stockage en mémoire stable.

La vision externe du modèle est basée sur la notion d'agents mobiles que nous appelons *assistants*. L'interface MAP permet d'activer un assistant pour qu'il exécute un ensemble d'opérations sur des nœuds distants, et à tout moment de récupérer le résultat du travail d'un assistant ou de détruire l'assistant. De manière interne, le modèle est constitué d'un support d'exécution des assistants qui permet d'interpréter le programme d'un assistant, de migrer l'exécution d'un assistant vers un nœud distant, d'instancier des clones d'un assistant et d'envoyer des résultats.

La conception de ce modèle a été influencée par les caractéristiques des réseaux de grande envergure comme l'Internet :

- pour permettre les opérations déconnectées, un assistant est un processus asynchrone : l'application active un assistant et collecte les résultats de son travail plus tard,
- pour réduire le trafic réseau, un assistant déplace son exécution sur des nœuds distants pour exécuter des opérations sur les données locales,
- pour prendre en compte le nombre important de nœuds, un assistant instancie des clones qui s'exécutent en parallèle comme des entités autonomes,
- pour palier les pannes ou les arrêts de nœud, un assistant est un processus persistant : l'état d'exécution est sauvegardé en mémoire stable et peut être restauré aprés redémarrage.

L'exemple motivant la conception de cette architechture est présenté sur la figure 1. L'utilisateur active un assistant sur un nœud MAP en demandant le stockage des résultats sur un nœud proche de Boston, se déconnecte et fait un trajet jusqu'à Boston. Pendant ce temps, l'assistant effectue un travail utile en collectant de l'information sur différents nœuds dans le réseau. Les résultats sont ensuite récupérés par l'utilisateur qui se connecte à Boston.

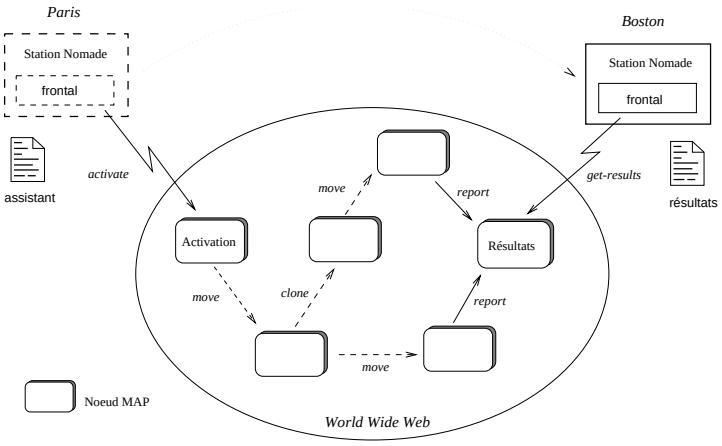

Fig 1 : Une application sur le réseau MAP

La figure 2 illustre les principes du modèle MAP. Une application active un assistant sur un nœud distant pour réaliser une tâche complexe et se déconnecte. Pour accomplir sa tâche l'assistant accède à l'information et aux services puis se déplace sur un nœud distant et instancie un clone sur un autre nœud. Les deux exemplaires de l'assistant accèdent à l'information et aux services en parallèle et produisent des résultats. Ces résultats sont conservés dans un collecteur de résultats qui est un objet persistant unique pour une activation. A tout moment l'application peut se connecter et récupérer les résultats auprès de ce collecteur. Sur chaque nœud l'interpréteur est chargé de l'exécution du programme d'un assistant. Il assure la sauvegarde de l'état d'exécution de l'assistant à chaque opération critique, fournissant ainsi un point de reprise en cas de pannes. Aprés une panne tous les assistants sont restaurés par le système..

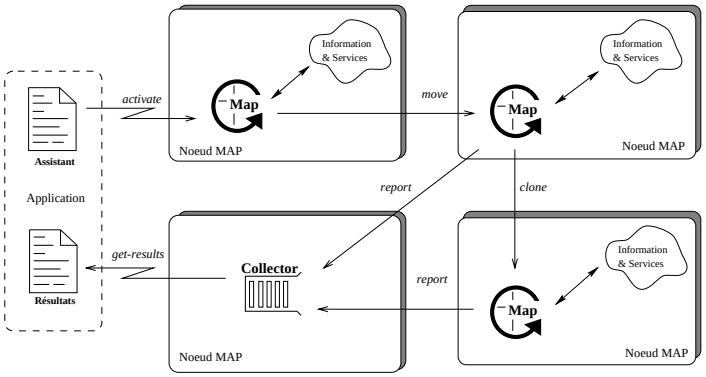

Fig 2 : Les principes du modèle MAP

La figure 3 présente l'architecture fonctionnelle du modèle composée de quatre couches : le *Frontal* d'application réalise l'interface applicative en utilisant les assistants ; la couche des *Assistants* réalise l'interface externe des assistants ; la couche des *Primitives MAP* réalise les opérations internes du modèle disponibles pour le code des assistants ; la couche du *Support système* réalise les fonctions de sauvegarde et de restauration d'état d'exécution et les fonctions de communication..

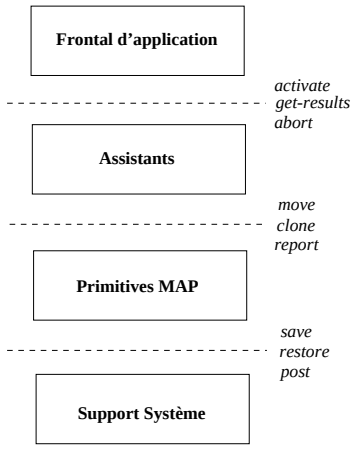

Fig 3 : L'architecture fonctionnelle de MAP

2.1 Interface externe de MAP

Le frontal d'application dispose de l'interface MAP pour l'activation et le contrôle des assistants, ainsi que pour la récupération des résultats :

- **`activate`** *source-code activate-node result-node*
  La procédure **`activate`** initialise l'exécution d'un assistant avec le programme fourni par l'application et crée un collecteur de résultats. Elle retourne une capacité sur l'assistant pour utilisation ultérieure.

- **`get-results`** *assistant-capability*
  La procédure **`get-results`** utilise la capacité sur l'assistant pour récupérer les résultats de l'exécution auprès du collecteur. Cette procédure détecte la terminaison de l'assistant et de ses clones. La détection de la terminaison est basée sur l'analyse du graphe de clonage de l'assistant construit à partir des informations de terminaison de chaque processus : l'identité de son créateur et le nombre de clones qu'il a instancié. Elle retourne l'état de terminaison et les résultats courants. Si l'exécution est terminée, le collecteur est détruit.

- **`abort`** *assistant-capability*

son de l'assistant et de ses clones. Si l'exécution n'est pas terminée, elle lance une procédure de destruction d'exécution répartie qui force l'assistant et ses clones à se terminer immédiatement. Cette procédure repose sur deux mécanismes :

- le collecteur n'accepte plus de rapports et force la terminaison des émetteurs lors de la communication ;
- un signal de terminaison de l'assistant et de ses clones est envoyé à une liste de nœuds qui sont chargés de propager ce signal selon les informations de migration et de clonage qu'ils détiennent. Cette liste est composée du nœud d'activation et des nœuds d'où les rapports ont été émis.

On retourne l'état de terminaison au moment de l'appel. Le frontal d'application devra utiliser la procédure **`get-results`** pour récupérer les résultats et avoir la confirmation de la terminaison.

2.2 Primitives d'assistants mobiles

Les assistants disposent des primitives pour effectuer certaines opérations au cours de leur exécution. Ces primitives concernent la mobilité, le clonage, et le renvoi de résultats :

- **`move`** *node*
  La procédure **`move`** transfère un assistant vers un nœud distant. L'état d'exécution du processus est sauvegardé, puis envoyé sur le nœud distant où il est restauré. Cette procédure utilise un schéma d'authentification pour vérifier si l'assistant est autorisé à s'exécuter sur le nœud cible. Cette procédure s'exécute de façon atomique et retourne un statut.

- **`clone`** *node id*
  La procédure **`clone`** crée une copie de l'assistant avec une identité nouvelle et transfère le clone de l'assistant vers un nœud distant. Le schéma utilisé est le même que pour la procédure de migration. Cette procédure s'exécute de façon atomique et retourne un statut.

- **`report`** *message*
  La procédure **`report`** envoi le message vers le collecteur de résultats. On ajoute de l'information sur l'état de l'assistant au message qui est acheminé sur le nœud du collecteur. Cette procédure s'exécute de façon atomique et retourne un statut.

- **`break`** *exception*
  La procédure **`break`** stoppe l'exécution de l'assistant ; l'information sur les clones qu'il a instancié est ajoutée au message d'exception envoyé au collecteur.

- **`exit`**
  La procédure **`exit`** stoppe l'exécution de l'assistant, l'information sur les clones qu'il a instancié est envoyé au collecteur.

- **`node`**
  La fonction **`node`** retourne l'identificateur du nœud courant.

- **`identity`**
  La fonction **`identity`** retourne l'identificateur de l'assistant courant.

2.3 Support système

L'exécution des assistants nécessite un support système pour la sauvegarde et la restrauration de l'état d'exécution, ainsi que le transfert de données. Ces fonctions sont internes au système MAP :

- **`save`**
  La procédure **`save`** crée un point de reprise en sauvegardant l'état d'exécution de l'assistant.

- **`restore`**
  La procédure **`restore`** reprend l'exécution de l'assistant à partir d'un point de reprise.

- **`post`**
  La procédure **`post`** permet le transfert atomique des données avec authentification. Elle prend en compte les problèmes de rupture de communication par des techniques transactionnelles (estampilles, journal, retransmissions) et utilise un schéma d'authentification basé sur la signature digitale.

2.4 Discussion

Dans le modèle MAP, les assistants sont des entités indépendantes dont le comportement est asynchrone. Un assistant et ses éventuels clones s'exécutent en parallèle indépendament les uns des autres. La persistance de l'exécution des assistants repose sur la création de points de reprise en mémoire stable. Lorsqu'un assistant exécute une primitive du modèle qui modifie son environnement, à savoir *move*, *clone* et *report*, le système crée un point de reprise. La primitive et le point de reprise sont réalisés comme une action atomique ce qui permet d'assurer qu'un assistant est toujours redémarré dans un état cohérent. Ce mécanisme permet de faire progresser l'exécution des assistants malgré les pannes de nœud. La communication entre des nœuds MAP à l'aide de la procédure *post* garantit des résultats corrects en présence des pannes de sites et des coupures de connexion. Néanmoins, la communication n'est pas optimisée pour des connexions à bas débit.

Nous avons développé un prototype du modèle MAP dans le contexte du World-Wide Web en choisissant Scheme comme langage pour la programmation des assistants. Scheme est un langage à part entière basé sur le *lambda-calcul* : une procédure est un élément du langage au même titre qu'une variable ou une constante, il n'y a pas de séparation entre le code et les données. Le programme d'un assistant est alors défini par une liste d'expressions Scheme. Nous utilisons le WWW comme support de communication au travers du protocole HTTP et comme support d'exécution distribué via le mécanisme d'extension CGI. La figure 4 illustre les principes d'implémentation du modèle MAP. .

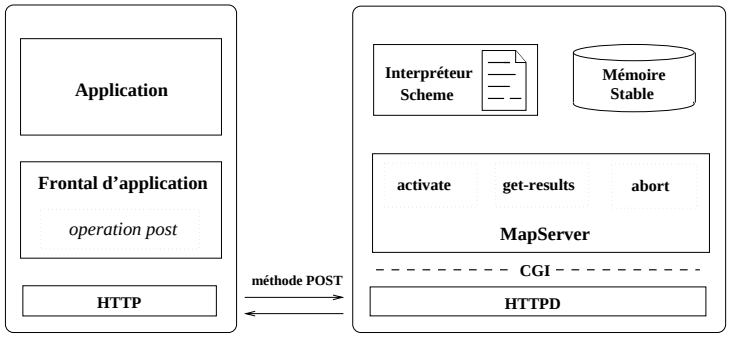

Fig 4 : Les principes d'implémentation de MAP

Chaque nœud qui participe au réseau des assistants comporte un serveur HTTP auquel on ajoute un programme CGI nommé *MapServer*. Ce programme dispose d'un interpréteur Scheme et d'une mémoire stable, il implante toutes les fonctions du modèle qui sont rendues accessibles aux assistants via l'interpréteur. Chaque assistant est un processus exécutant l'interpréteur qui a été modifié :

- en otant les primitives d'accés direct au système pour une meilleure sécurité,
- en ajoutant les primitives du modèle MAP,
- en ajoutant des primitives d'accés au WWW pour l'accés aux documents et des services WWW,
- en ajoutant les fonctions de sauvegarde et de restauration de l'état d'exécution de l'interpréteur.

La communication entre les nœuds est assurée par la fonction interne **`post`** qui permet de transfèrer atomiquement des données avec authentification. Cette fonction est implantée en utilisant la méthode POST du protocole HTTP. La cible de ce postage est le progamme CGI *MapServer* qui doit réaliser l'authentification et prendre en compte les ruptures de communication. Les données du postage peuvent être soit le

un signal de contrôle. Dans tous les cas les données du postage font l'objet d'une signature digitale à l'aide de MD5 et d'une clef secrète.

2.6 Exemple d'application MAP

Nous avons développé une application d'accés à l'information répartie sur le WWW pour les stations nomades en utilisant le prototype du modèle MAP [PERRET96D]. Cette application baptisée*WebFinder* propose la recherche de documents par requête sur un ensemble de serveurs WWW. Par rapport à un robot classique qui charge tous les documents sur un site central, on réduit le volume de données transmises, améliore les temps de réponse et offre la possibilité de se déconnecter durant la recherche.

3. Architecture nomade

La problématique des stations nomades et de la communication mobile devient une direction importante des recherches actuelles en réseaux. Le point de départ est la constatation des caractéristiques suivantes :

- les stations nomades disposent d'une mobilité soit en utilisant des connexions sans-fil, soit en se déplaçant d'un point fixe d'accés au réseau à un autre,
- les stations nomades communiquent au travers de réseaux hétérogènes dont le débit, la latence, le taux d'erreurs et le coût sont trés variés,
- les ressources réseau sont critiques, souvent à un débit bas (connexions sans-fil) et doivent être optimisées,
- une station nomade change souvent son état en passant d'un état de marche à l'état d'arrêt, ou d'un état connecté à l'état déconnecté,
- le mode de travail synchrone où la station cliente est active pendant la durée de traitement sur le serveur n'est pas adapté aux stations nomades ; le mode asynchrone où l'application délègue le travail à accomplir aux stations du réseau peut apporter une amélioration des performances perçus par l'utilisateur,
- l'accés à l'information répartie devient une application pilote pour les stations nomades.

La mobilité des stations nomades doit être assurée de manière indépendante de la localisation. Cette indépendence peut être obtenue au niveau réseau à l'aide d'une couche de mobilité IP appelée *IP mobile* pour prendre en compte les déplacements de stations [PERKINS96,CHESHIRE96]. Cette solution permet d'acheminer des paquets IP vers une station qui se connecte à un réseau quelconque par l'intermédiaire des routeurs de mobilité placés sur le réseau d'origine de la station nomade et sur le réseau visité. Le routeur sur le réseau d'origine utilise un tunnel IP pour transmettre des paquets au routeur sur le réseau visité qui se charge de les transmettre à la station nomade. De cette façon, une station peut garder des connexions ouvertes sur son réseau d'origine et les utiliser sur le réseau visité.

La communication durant un déplacement doit faire appel aux connexions sans-fil comme par exemple GSM ou des réseaux locaux sans-fil. La qualité de transmission est variable selon l'endroit où se trouve la station nomade. Le coût élevé de ce type de connexion nécessite l'optimisation de leur utilisation.

Les problèmes de mobilité et des connexions intermittentes sur des réseaux hétérogènes font qu'une station nomade a besoin d'une couche intermédiaire qui cache les particularités des connexions réseau et permet de les gérer au mieux. Une structuration de protocoles pour les stations nomades a été proposée par Kleinrock [KLEINROCK95A,KLEINROCK95] :

- une couche des réseaux ouverts de données (*Open Data Network*) qui utilisent différentes technologies de transmission de base ; à ce niveau apparaît le support de nomadicité - des routeurs nomades,
- une couche des services de transport (*Transport Services*),
- une couche intermédiaire des services (*Middleware*) qui offrent un support pour des applications nomades,
- une couche d'application.

La couche intermédiaire *middleware* proposée par Kleinrock présente une interface MIMI (*Middleware/Middleware Interface*) qui fourni des services spécifiques aux applications nomades :

- agents autonomes (pour exécuter des actions dans le réseau),
- cache prédictif (pour précharger des données sur la station nomade),
- approximation des résultats (pour fournir des résultats partiels ou approchés et adapter la qualité de service en fonction des ressources réseaux disponibles),
- gestion de connexion (pour gérer les connexions et les déconnexions ainsi que les arrêts et les démarrages de la station),
- services de sécurité (pour authentifier et autoriser l'accés),
- gestion de performances (pour informer les applications sur l'état des ressources et conseiller leur utilisation optimale),
- gestion de localisation (pour aider à localiser des services dans le réseau),

L'utilisation de cette interface suppose que les applications nomades deviennent ”conscientes” de la mobilité et de certains problèmes liés à la communication. Par exemple, l'application peut vouloir négocier la qualité de service en fonction des connexions disponibles ou demander des résultats partiels ou approchés si les résultats complets sont indisponibles ou s'ils impliquent trop de ressources. L'application peut aussi travailler de manière adaptative - elle peut se baser sur une prédiction d'utilisation des ressources réseau et ajuster son comportement en fonction des informations venant du réseau.

Une architecture pour des stations nomades doit aussi prendre en compte le type d'applications visées. On peut en identifier trois qui sont de première importance ou le seront dans l'avenir [MESSERSCHMITT96] :

- le filtrage de l'information : il est important de pouvoir trier de l'information disponible sur le réseau et de ne choisir que l'information pertinente,
- le filtrage des communications asynchrones : nous sommes de plus en plus submergés par le traitement des communications asynchrones comme par exemple le courrier électronique ; il devient important de filtrer les messages, d'associer des priorités et de traiter les messages en fonctions de ces priorités,
- la négociation des communications synchrones : il est parfois plus efficace d'arranger une communication synchrone (par exemple une session de vidéo-conférence) que de communiquer de manière asynchrone.

Notre objectif est de considérer ces types d'applications dans un environnement nomade. On peut constater que le paradigme basé sur des agents mobiles peut apporter beaucoup d'avantages à une architecture présentant l'interface MIMI. Le paradigme apporte le mode asynchrone et permet un gain de performances significatif. Autour de ce paradigme, on peut construire d'autres fonctions correspondant aux services spécifiques pour des applications nomades présentés plus haut. Nous proposons donc d'ajouter à l'architecture d'agents mobiles MAP une couche d'adaptation nomade qui élargira les fonctions MAP proposées aux applications. Cette couche permettra de gérer le collecteur des résultats comme un cache prédictif, de fournir des résultats partiels ou approchés et d'optimiser l'utilisation de ressources réseau critiques.

Nous supposons que la couche d'adaptation nomade dispose de connexions appropriées au niveau des réseaux ouverts de données (comme par exemple la solution IP mobile).

3.1. Couche d'adaptation nomade pour MAP

Le modèle MAP offre le support d'agents mobiles, à savoir l'activation des assistants et la récupération des résultats. Pour prendre en compte des connexions réseau intermittentes dont les caractéristiques sont variables et souvent critiques, il nous faut ajouter des fonctionnalités supplémentaires de gestion et d'optimisation de connexions.

Nous avons considéré deux approches pour adapter l'architecture MAP aux stations nomades. La première consiste à considérer une station nomade comme un nœud MAP et d'ajouter les fonctions d'optimisation des connexions à la fonction *post* du support système MAP. Une application pourrait activer des assistants sur une station nomade et les transferts vers d'autres nœuds MAP seraient optimisés. Cette solution traite tous les nœuds de manière homogéne ce qui probablement n'est pas souhaitable - le code d'optimisation des connexions serait exécuté pour tous les transferts, même sur des connexions qui n'ont pas besoin de cette fonctionnalité.

La deuxième approche traite une station nomade de manière spécifique où elle communique avec un nœud MAP stationnaire par un protocole d'optimisation des connexions. Cette approche place des fonctions supplémentaires là où elles sont

nœuds MAP et allége l'architecture du coté de la station nomade. Pour ces raisons, nous avons choisi cette deuxième approche.

L'ajout des fonctions supplémentaires dans l'interface ouverte aux applications signifie qu'elles deviennent “conscientes de la nomadicité”. Par exemple, elles pourront recevoir des notifications signalant l'état des connexions pour s'adapter aux conditions variables.

La figure 5 présente l'architecture MAP avec la couche d'adaptation nomade. Sur une station nomade, cette couche présente l'interface de l'activation des assistants, de la récupération des résultats et de la gestion des connexions.

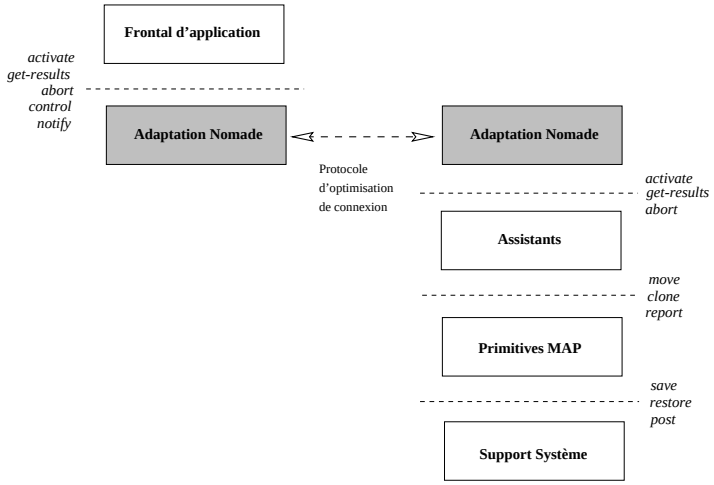

Fig 5 : L'architecture fonctionnelle nomade

La figure 6 illustre les principes de la couche d'adaptation nomade. Il y a trois types d'entités : une station nomade, une base MAP et des nœuds MAP. Une station nomade coopère avec une base MAP à l'aide d'un protocole d'optimisation de connexions. Pour les applications sur une station nomade, la couche d'adaptation nomade agit comme un intermédiaire pour l'activation des assistants et la récupération des résultats et rend visible le collecteur des résultats. Le collecteur est géré comme un cache prédictif. L'application informe la couche sur les politiques à adopter pour la gestion de connexion et sur les opérations à effectuer sur les résultats. La couche d'adaptation nomade notifie l'application sur l'état de connexion.
L'interface MAP, en plus des fonctions d'activation des assistants et de récupération des résultats propose les fonctions de gestion des connexions suivantes :

- **`control`** *attribute profile code*

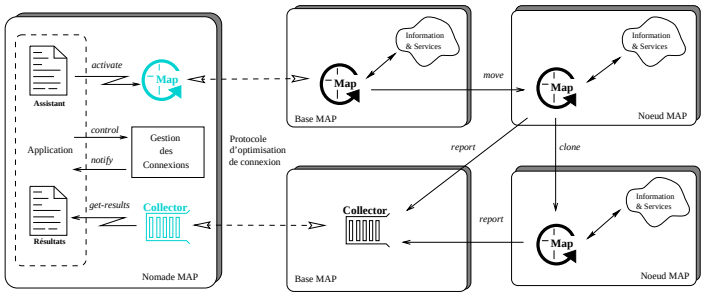

Fig 6 : Les principes de l'architecture nomade

La procédure **`control`** passe au gérant de connexions le profil concernant un *attribut* de connexion. Le *profil* spécifie comment l'attribut de connexion doit être optimisé. Par exemple, pour une connexion GSM, l'application peut spécifier que le gérant doit optimiser le coût de connexion en diminuant le volume de données à transfèrer et en groupant des données. L'application peut aussi spécifier comment des résultats doivent être traités au cours de transfert sur la station nomade. Par exemple, elle pourra préciser que des données du type MIME **`image/jpeg`** doivent être dégradées en qualité par la diminution de la résolution. L'application peut fournir du *code* de traitement des données à transfèrer ainsi que spécifier comment faire une approximation des résultats ou comment obtenir des résultats partiels.

- **`notify`** *attribute value*
  La procédure **`notify`** présente de l'information concernant un attribut de connexion.

Le gérant de connexion coopère avec la base MAP à l'aide d'un protocole d'optimisation de connexion. Ce protocole maintient de l'information sur les attributs de connexions et agit en fonction de la spécification de contrôle exprimée par l'application. Le protocole peut effectuer un traitement des données sur la station nomade ou sur la base MAP avant le transfert. Ce traitement peut être la compression de données, la dégradation de la qualité ou une opération quelconque dont le code sera fourni par l'application.

Les profils de connexion donnent des indications sur les politiques à adopter pour l'optimisation des connexions. Ils spécifient le critère de coût ou des opérations à réaliser sur des données.

Le gérant de connexion met en œuvre un cache prédictif du collecteur. Il organise le transfert du collecteur sur la station nomade en optimisant la connexion. Le transfert est une suite d'actions atomiques avec une sémantique transactionnelle - son effet est correct en présence des pannes ou des arrêts de sites et des coupures de connexion.

veau de ressources consommées (p.ex. le temps de connexion) et de savoir quelle partie des résultats est déjà transférée sur la station nomade.

3.2 Etat d'avancement

L'architecture MAP a été mis en œuvre et un prototype est opérationnel. Il a été testé avec une application de recherche d'information *WebFinder*. Nous sommes en train d'implémenter la couche d'adaptation nomade en utilisant dans un premier temps un portable et une connexion GSM. L'application pilote qui va servir de test au prototype est l'accés optimisé au WWW. L'utilisateur pourra donner par exemple une liste d'URL et spécifier une requête comportant des mots clefs, la date de mise à jour des documents, et la profondeur de parcours des ancres hypertextes. L'application active des assistants pour parcourir l'espace de documents à partir de la liste d'URL et ramène tous les documents correspondants à la requête sur la station nomade. Ensuite l'utilisateur peut consulter efficacement ces documents. L'utilisateur pourra aussi spécifier quel traitement il faut associer aux données volumineuses, par exemple comment dégrader la qualité des images ou de la vidéo.

4. Conclusions

Nous avons présenté une adaptation de l'architecture d'agents mobiles MAP à l'environnement nomade. L'approche basée sur les agents mobiles apporte plusieurs avantages par rapport aux approches traditionnelles. Elle permet le mode de travail asynchrone et la programmation d'applications par délégation. La couche d'adaptation nomade gére le collecteur des résultats comme un cache prédictif, fournit des résultats partiels ou approchés et ajoute au modèle MAP la gestion optimisée des connexions.

La prochaine étape de notre travail est la prise en compte des applications de négociation des communications synchrones. Nous voudrions utiliser des assistants pour négocier des rendez-vous : des assistants se déplacent sur des nœuds concernés et initialisent des communications synchrones multimédia, par exemple une session de vidéo-conférence.

Bibliographie

[BERNERS-LEE92] T. BERNERS-LEE ET AL. *World-Wide Web: The information universe*. Electronic Networking, 2(1):52--58, 1992.

[CHESHIRE96] S. CHESHIRE M. BAKER. *Internet mobility 4x4*. ACM Computer Communication Review, 26(4), 1996.

[KIISKINEN96] J. KIISKINEN, ET AL. *Data channel service for wireless telephone links*. IEEE Bulletin on Operating Systems and Application Environments, 8(1):3--12, 1996.

[KLEINROCK95A] L. KLEINROCK. *ARPA PI meeting presentation*. URL: http://millennium.cs.ucla.edu/LK/lkpimtgfla795/index.html, 1995.

[KLEINROCK95B] L. KLEINROCK. *Nomadic computing-an opportunity*. ACM Computer Communication Review, 25(1):36--40, 1995.

[MESSERSCHMITT96] D.G. MESSERSCHMITT. *The convergence of communications and computing: What are the implications today?* IEEE Proceedings, août 1996.

[PERKINS96] C.E. Perkins. *IP mobility support*. draft-ietf-mobileip-protocol-16.txt, 1996.

[PERRET96A] S. PERRET A. DUDA. *Design and implementation of MAP: A system for mobile assistant programming*. Proc. IEEE International Conference on Parallel and Distributed Systems, Tokyo, juin 1996.

[PERRET96B] S. PERRET A. DUDA. MAP: *Mobile assistant programming for large scale communication networks*. Proc. IEEE International Communications Conference 96, Dallas, juin 1996.

[PERRET96D] S. PERRET A. DUDA. *Mobile assistant programming for efficient information access on the WWW*. Proc. Fifth International World-Wide Web Conference, Paris, mai 1996.