Découverte de services sur un réseau de petits ordinateurs

Service discovery on networks of small computers

François Lévesque[*] et Michel Barbeau[**]

[*]*Centre de recherches sur les communications*
3701, avenue Carling, C.P. 11490, succursale H
Nepean (Ontario) Canada K2H 8S2

[**]*School of Computer Science*
Carleton University
5302 Herzberg Laboratories
1125 Colonel By Drive
Ottawa (Ontario) Canada K1S 5B6

RÉSUMÉ. Cet article porte sur les problèmes liés à la découverte de services sur un assistant numérique personnel (ANP). Ceci rend possible la configuration dynamique des applications afin qu'elles accèdent aux ressources du réseau auquel l'ANP est connecté. Nous avons réalisé un système de gestion et de découverte de services à partir d'un ANP. L'architecture de ce système est répartie. Les ressources sont abstraites sous forme de composants répartis CORBA. Cette approche rend les applications indépendantes de l'implantation des services. Elle rend aussi uniforme les interfaces d'accès aux ressources.

ABSTRACT. The research presented in this paper is about the problem of discovery of services from a Personal Digital Assistant (PDA). A solution to this problem makes possible automatic configuration of applications such that they can access resources over networks to which they are attached. We developed a system for service management and service discovery from an PDA. The architecture of this system is distributed. Resources are abstracted as CORBA distributed components. This approach makes application independent of service implementation and makes uniform interfaces to resources.

MOTS-CLÉS : CORBA, création de services, gestion de services, découverte automatique de services, réseau mobile, assistant numérique personnel, composant réparti.

KEY WORDS: CORBA, service creation, service management, automatic service discovery, mobile network, personal digital assistant, distributed component.

Calculateurs parallèles, Systèmes répartis et Réseaux.

1. Introduction

La notion de réseau du futur évoque dans l'esprit des gens des idées telles que la fibre optique, l'accès à Internet à débit élevé, la transmission de l'audio et de la vidéo et le commerce électronique. Le nombre de stations de travail exploitant ces idées va certainement continuer à croître de façon exponentielle. En parallèle à cela, il existe une autre catégorie de processeurs qui feront vraisemblablement partie du réseau du futur, c'est-à-dire les processeurs embarqués. Alors que le nombre de stations de travail se compte en millions, celui des processeurs embarqués peut se compter en milliards. Ils sont présents dans les montres, les appareils électroménagers, les appareils de chauffage, les autos, etc. Le branchement au réseau de ces petits processeurs va rendre possible la surveillance et le contrôle à distance des équipements dans lesquels ils sont embarqués.

Considérons à titre d'exemple la situation illustrée à la figure 1. Présentement, lorsque Charles désire utiliser son auto, il doit faire fonctionner deux dispositifs de contrôle à distance, le premier actionne l'ouverture de la porte du garage alors que le second désengage le système antivol de son véhicule (en haut à droite). Supposons que les processeurs embarqués contrôlant la porte du garage et le système antivol soient équipés d'interfaces sans fils et qu'ils exécutent des protocoles réseau leur permettant d'annoncer des services, d'être découverts et de communiquer les uns avec les autres. De plus, supposons que Charles ait un assistant numérique personnel (ANP) sans fils équipé de protocoles de réseau ad hoc ou mobile et de découverte dynamique de services (en haut à gauche). Ainsi, Charles ouvrirait son ANP et activerait un mécanisme de découverte de services le conduisant à obtenir l'accès aux services *ouvrir la porte du garage* et *désengager le système antivol* (en bas à gauche). Charles déclencherait ces services de son ANP, obtiendrait l'accès à son auto et serait prêt à partir (en bas à droite).

L'interconnexion d'ANP et de processeurs embarqués soulève une question. Comment ces petits processeurs peuvent-ils prendre connaissance les uns des autres et communiquer ? Bien sûr, ce serait bien si cela pouvait se faire sans fils et d'une manière automatique. Idéalement, à la figure 1, l'ANP découvrirait dynamiquement avec peu de configuration la porte du garage et le système antivol.

La réponse à cette question repose sur le développement de technologies matérielles (ANP, processeurs embarqués, interfaces sans fils, etc.) et logicielles, telles que les protocoles de réseaux mobiles, les protocoles de réseaux ad hoc, les protocoles de découverte de services, les composants répartis, le code mobile, etc.

Cet article se concentre sur le problème de la découverte de services dans un tel environnement. Il présente l'architecture d'un logiciel original pour la création de services sur un réseau et la découverte de services sur un ANP. Notre solution repose sur l'architecture de composants répartis CORBA. Notre système permet à un administrateur d'un réseau d'inscrire, de modifier et de retirer des ressources matérielles ou logicielles disponibles sur le réseau. Les applications spécifient les services dont elles ont besoin puis les ressources du réseau correspondant à ces services sont découvertes dynamiquement à l'aide de notre système.

Ce genre de solutions exige parfois des protocoles spécifiques pour communiquer avec les ressources découvertes. Cela implique qu'un grand nombre de protocoles doit être implanté sur l'ANP. Notre solution a l'avantage d'éliminer le besoin d'avoir tous ces protocoles sur un ANP puisque nous utilisons des composants répartis. Ainsi, aucun code spécifique n'est nécessaire aux applications pour communiquer avec les services. Les composants répartis offrent une interface uniforme et dissimulent les détails d'implantation.

La section 2 passe en revue les technologies logicielles associées à la découverte de services sur un réseau. La section 3 introduit les composants répartis et CORBA. La section 4 présente notre réalisation. La section 5 contient une discussion. Nous concluons avec la section 6.

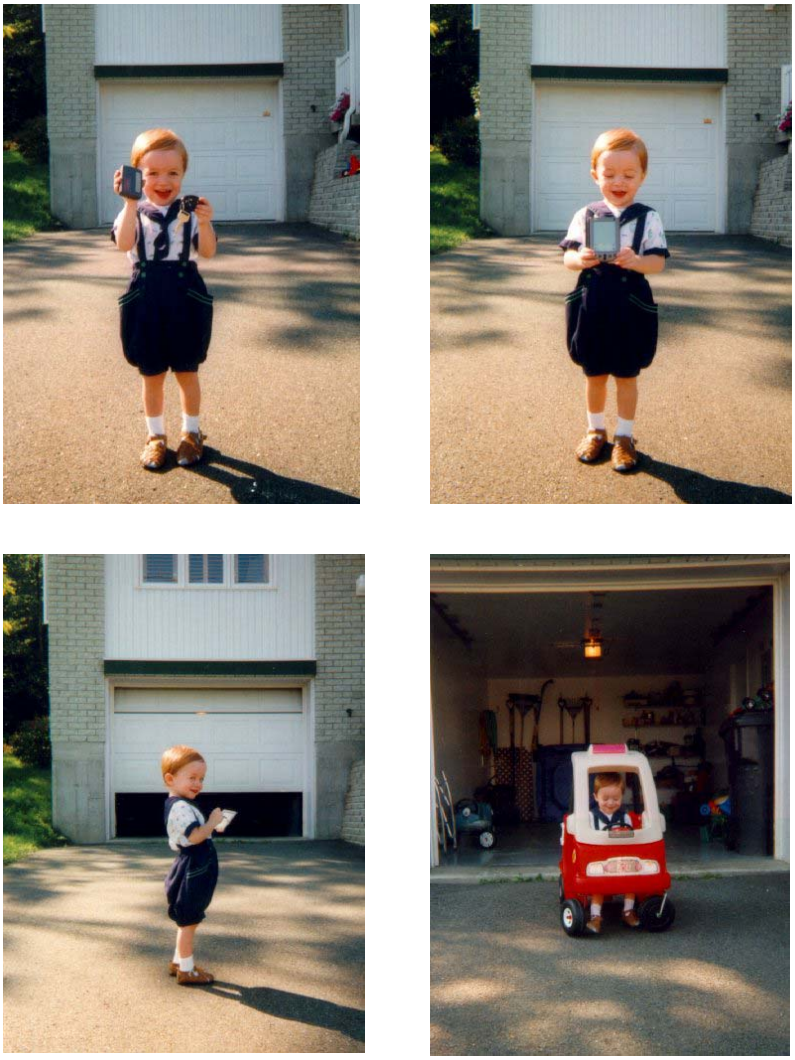

Figure1. Scénario typique.

2. La découverte automatique de services

Un *service* est une action accomplie par une ressource d'un réseau d'ordinateurs. Celle-ci est soit matérielle, soit logicielle, soit les deux. De plus, un service est accessible sur le réseau au moyen d'un protocole d'application.

Les services font l'objet de processus administratifs et de découverte. L'administration des services implique, entre autres, la définition des ressources, la spécification des droits d'accès et la configuration des services. Le processus de découverte amène les applications à prendre connaissance de la disponibilité de services. Il s'effectue au moyen de protocoles de télécommunications spécialisés. Ceux-ci sont employés, entre autres, par les applications sur ordinateurs mobiles qui se déplacent et se branchent aux réseaux. L'information nécessaire pour contacter un service s'appelle le *point d'accès aux service*. Les *protocoles de découverte automatique de services* permettent aux applications de s'auto configurer et d'obtenir dynamiquement des points d'accès aux services dont elles ont besoin pour fonctionner.

La problématique de la découverte automatique de services est bien introduite dans l'ouvrage de Kempf et St. Pierre [KEM 99]. En général, ce processus fait intervenir à des degrés divers trois type d'acteurs : des agents de service, des agents d'utilisateur et des agents de répertoire. Les *agents de service* représentent les services et font connaître leur existence. Les *agents d'utilisateur* agissent aux noms des clients des services et s'enquièrent des points d'accès au service. Les *agents de répertoire* agissent comme intermédiaires entre les agents de service et les agents d'utilisateur.

Une annonce de la présence d'un service comporte potentiellement trois éléments d'information : un nom de type de service, une liste d'attributs descriptifs (dont le nom du protocole d'application à utiliser pour communiquer avec le service) et le point d'accès au service.

Kempf et St. Pierre ont identifié quatre modèles de découverte dynamique de services : i) le contact d'agents de répertoire et la consultation de fichiers locaux, ii) le contact d'agents de service uniquement, iii) le contact d'agents de répertoire uniquement et iv) le contact d'agents de service et d'agents de répertoire.

2.1. Contact d'agents de répertoire et consultation de fichiers locaux

Le modèle de contact d'agents de répertoire et de consultation de fichiers locaux est largement employé sur l'Internet. Le point d'accès ultime à un service est une adresse IP et un numéro de port. Les services portent des noms standards (ex. : *www* pour les serveurs Web). Un programme client (l'agent d'utilisateur) entre en contact avec un DNS (l'agent de répertoire) pour obtenir l'adresse IP d'un serveur (les agents de service) localisé sur un réseau donné. Sous Unix, le programme client consulte le fichier */etc/services* pour obtenir le numéro de port associé au serveur et le nom du protocole transport (c'est-à-dire, UDP ou TCP) à employer pour communiquer.

Côté auto configuration, la communauté Internet a développé le *Dynamic Host Configuration Protocol* (DHCP) qui permet à un ordinateur nouveau sur un réseau de requérir et d'obtenir une configuration lui permettant de communiquer [ALE 97, DRO 97]. Au travers de DHCP, l'ordinateur obtient dynamiquement une adresse IP, un masque réseau, l'adresse d'un routeur, l'adresse d'un DNS et des adresses IP de serveurs.

Pour obtenir les données de configuration, un client doit d'abord découvrir un serveur DHCP par diffusion locale d'un message. Un ou plusieurs serveurs DHCP répondent au client par des offres de configuration. Le client accepte une offre, ignore les autres et demande au serveur choisi de lui affecter la configuration. Le processus est complété lorsque le serveur acquitte la demande du client. Avec le DNS et DHCP, les services sont identifiés à des adresses réseau et des numéros de port. Avec le DNS, la recherche de services est de type page blanche, c'est-à-dire par nom. Par ailleurs, la mise à jour, l'ajout, le retrait et la modification de point d'accès à des serveurs requièrent des modifications à deux endroits, chez le répertoire et dans le fichier local.

2.2. Contact d'agents de service uniquement

Un modèle de type contact d'agents de service uniquement est employé par les réseaux Appletalk qui sont de taille petite à moyenne, un intra réseau par exemple. Les réseaux sont constitués de sous réseaux, appelé zones, avec des routeurs entre chacun d'eux. L'approche fait intervenir deux protocoles : le *Name Binding Protocol* (NBP) et le *Zone Information Protocol* (ZIP) [SID 90]. À l'intérieur d'un sous réseau, des requêtes de service sont diffusées par des agents d'utilisateur dans le format du protocole NBP. Les agents de service, qui sont en mesure de le faire, répondent directement à la requête. Les requêtes peuvent se faire par type de service ou par caractéristiques de service recherché. Le protocole ZIP sert à propager d'un sous réseau à un autre l'information sur l'existence des zones. Un agent d'utilisateur peut diriger une requête vers une autre zone en employant des routeurs comme relais. Dans ce cas également, les agents de service répondent directement. Pour Appletalk, le point d'accès à un service est une adresse réseau et un numéro de port.

Le *Internet Control Message Protocol* (ICMP) comporte un mécanisme pour la découverte automatique des adresses IP des passerelles locales [DEE 91]. Les passerelles diffusent sur une base régulière des messages d'annonce ICMP contenant leurs adresses. Ces messages sont captés par les ordinateurs qui en utilisent le contenu pour se configurer. Les ordinateurs peuvent également solliciter explicitement de telles annonces de la part des passerelles en diffusant un message de requête. Ce mécanisme a été étendu par IP mobile pour la découverte automatique d'agents de mobilité [PER 96b].

Le protocole IPv6 comporte un mécanisme appelé *Neighbor Discovery* pour la découverte automatique des passerelles locales et l'acquisition automatique de certains paramètres de configuration, dont le *Maximum Transmission Unit* du réseau local [DEE 98] [NAR 98]. Ces mécanismes reposent sur la transmission régulière de

messages ICMP d'annonce par les passerelles et sur l'envoi de messages de sollicitation par les ordinateurs [CON 98], et est donc jusqu'ici analogue à celui présenté dans la référence [DEE 91]. Un mécanisme additionnel appelé *Neighbor Unreachability Detection* permet la détection automatique de défaillances de passerelle. Lorsque que cela survient l'ordinateur peut découvrir et utiliser une autre passerelle.

2.3. Contact d'agents de répertoire uniquement

Le modèle de type contact d'agents de répertoire uniquement est employé par le protocole de découverte de services (appelé le *browsing protocol*) de l'architecture *Common Internet File System* (CIFS) de Microsoft [LEA 97]. Cette architecture est créée dynamiquement par un processus d'élection.

Les réseaux sont divisés en domaines administratifs. Un domaine administratif est un groupe d'ordinateurs. Au moins une machine sur chaque domaine est désignée comme source d'information de service (l'agent de répertoire), on l'appelle le contrôleur maître. D'autres machines agissent comme contrôleurs secondaires. Le contrôleur maître reçoit les annonces de service (diffusées par des agents de service) et stocke les informations (avec durée de vie limitée) sur les services disponibles. Il passe ces informations aux contrôleurs secondaires sur demande et également sur une base périodique. Il n'interagit pas directement avec les agents d'utilisateur.

Les agents d'utilisateur transmettent des requêtes de découverte d'information de service aux contrôleurs secondaires, avec des appels de procédure à distance. Ils découvrent les contrôleurs secondaires par diffusion d'une requête. Le contrôleur maître répond par une liste de contrôleurs secondaires. Les requêtes de service sont formulées par type (ex. : un serveur Novell). Le point d'accès au service est un nom de machine. Cette approche est limitée à un nombre fixe de services (20) et donc peu extensible.

Côté IETF, *Lightweight Directory Access Protocol* (LDAP) est un autre protocole qui sert à administrer et retrouver des données stockées dans des répertoires [WAH 97]. Une entrée du répertoire représente une entité (ex. : une personne, une imprimante, un serveur). Le protocole LDAP offre plusieurs opérations pour administrer et rechercher des informations dans un répertoire : l'ajout, la modification ou le retrait d'une entrée, la recherche selon des critères et la comparaison d'une entrée avec une autre. La récolte automatique et le retrait d'information sur les services disponibles ne sont pas abordés par LDAP.

Pour la recherche d'information, l'agent d'utilisateur se connecte au serveur LDAP (l'agent de répertoire) sur un port standard. Dans sa requête, il spécifie la partie du répertoire dans laquelle la recherche doit être faite et quelles informations doivent être retournées. Cela implique que l'agent d'utilisateur doit connaître le schéma de répertoire. Un certain nombre de paramètres doivent être spécifiés dans une requête de recherche. Tout d'abord, l'agent d'utilisateur doit indiquer le *distinguished name* (DN) qui définit le point de départ de la recherche dans le *directory information tree* (DIT). Ensuite, il doit spécifier la portée de la recherche,

c'est-à-dire la profondeur que la recherche peut atteindre dans le DIT à partir du point de départ. L'agent d'utilisateur spécifie le filtre de recherche. Vient ensuite la liste d'attributs que le serveur doit retourner pour les entrées qui respectent les critères de recherche. Comme le protocole LDAP supporte les alias, l'agent d'utilisateur spécifie si les alias doivent être résolus ou non, ainsi que la limite quant au temps de recherche ou à la taille des résultats.

La raison d'être de LDAP n'est pas l'administration et la découverte de services, il peut toutefois être utilisé pour le faire. Une entrée peut représenter un type de service, ses attributs et le point d'accès au service. L'introduction d'un nouveau type de service implique la modification du schéma du répertoire et éventuellement une intervention humaine pour le faire. Le protocole LDAP est très répandu. Les utilisateurs de LDAP qui ne veulent pas s'incommoder d'un autre protocole de découverte de services peuvent utiliser dans une certaine mesure LDAP pour faire le travail.

La solution des réseaux NetWare de Novell repose sur le *NetWare Directory Services* (NDS), le *Service Advertising Protocol* (SAP) et le *NetWare Core Protocol* (NCP) [SAN 95]. Ce sont les serveurs (ex. : les serveurs de fichiers, les serveurs d'imprimantes et les serveurs d'applications) qui utilisent ce protocole pour publier des offres de service. Il leur permet d'annoncer par diffusion de messages leur présence au démarrage et leur future absence avant leur interruption.

Les agents de service diffusent périodiquement, en utilisant SAP, leur offre de service. Ces offres de service sont reçues et stockées, avec durée de vie, par le NDS (l'agent de répertoire). Les agents d'utilisateur transmettent des requêtes d'information sur les services au NDS au moyen des protocoles NCP et SAP et en utilisant le *unicast*. Le nombre de types de service disponibles est limité à quatre. Le point d'accès à un service est une adresse de machine et un numéro de port.

Le *Resource Discovery Protocol* (RDP) fourni aux utilisateurs mobiles un moyen léger et automatique de découvrir les ressources d'un réseau [PER 96a]. Un agent d'utilisateur s'adresse à un serveur RDP, en utilisant le *unicast*, pour requérir les ressources d'un réseau. L'obtention de l'adresse du serveur RDP ne fait pas partie du protocole. Le client peut l'obtenir via un fichier de configuration ou dynamiquement par DHCP.

Un agent d'utilisateur formule dans une requête une liste de *Uniform Resource Name* (URN) pour demander des ressources et obtient en réponse un ou plusieurs *Uniform Resource Locator* (URL) pour chacun d'eux. L'URL correspond au point d'accès au service. RDP supporte aussi l'enregistrement ou le retrait dynamique de ressources avec un modèle de communication requête-réponse.

JINI est une architecture Java qui permet de grouper des périphériques et des composants logiciels en fédération dans un système réparti dynamique [SUN 99]. JINI fournit des mécanismes pour la construction de services, la recherche de services, la communication avec les services et leur utilisation dans les systèmes répartis.

Les agents de répertoire sont appelés *Lookup Service*. Quatre protocoles sont intégrés dans l'architecture : le protocole de service, le protocole de découverte, le protocole d'adhésion et le protocole de recherche. Le protocole de service, un ensemble d'interfaces Java, est utilisé par les services pour communiquer entre eux. Le protocole de découverte permet aux agents de service de découvrir un *Lookup Service* après quoi ils peuvent se joindre au système en utilisant le protocole d'adhésion. Enfin, le protocole de recherche est utilisé par les agents d'utilisateur pour fouiller le *Lookup Service*.

Pour rendre un service utilisable, il doit être enregistré auprès d'un *Lookup Service*. Un ou plusieurs *Lookup Service* peuvent exister sur un réseau. Chaque *Lookup Service* est affecté à un ensemble de groupes appelés communauté [KEI 99]. Deux formes différentes du protocole de recherche sont disponibles. La première forme est utilisée lorsqu'un agent de service ne connaît pas de *Lookup Service*. Il y a deux types d'interaction : l'agent de service envoi des requêtes pour trouver des *Lookup Service* ou bien le *Lookup Service* envoi des messages pour s'annoncer. La seconde forme de découverte est utilisée par un agent de service pour contacter un *Lookup Service* particulier, lorsque son URL est connu.

La communication avec un *Lookup Service*, ou un agent de service, se fait au travers d'un objet mandataire. Lors de l'enregistrement avec le protocole d'adhésion l'agent de service fournit un objet mandataire pour le service. Cet objet mandataire est inséré dans le *Lookup Service* avec des attributs descriptifs, sous forme d'objet sérialisé ou de référence à un objet distant. L'objet mandataire contient une interface Java au service : des méthodes qui peuvent être appelées pour exécuter le service.

2.4. Contact d'agents de service et d'agents de répertoire

Service Location Protocol (SLP) est le protocole développé par l'IETF dans le but de découvrir les ressources et les services d'un réseau [VEI 97]. L'enregistrement et le retrait de services sont dynamiques. L'administration est répartie et non hiérarchique.

Ce protocole fait intervenir quatre types d'entités différentes : des agents d'utilisateur, des agents de service, des mandataires et des agents de répertoire. Un mandataire est fonctionnellement équivalent à un agent d'utilisateur ou un agent de service et les remplace dans les applications ou services qui sont incapables d'utiliser SLP.

Les ressources et les agents d'utilisateurs sont groupés en domaines administratifs. La portée d'une requête de recherche de services est limitée à un domaine administratif. La recherche se fait par type de service ou par caractéristiques. Le point d'accès à un service est un URL.

Un agent de répertoire peut être présent ou absent. Lorsqu'il y en a un, les agents de service enregistrent leur offre auprès de l'agent de répertoire alors que les agents d'utilisateur envoient des requêtes à l'agent de répertoire pour obtenir les points

d'accès aux services. L'enregistrement d'un service est décrit par des valeurs d'attribut et se voit affecter une durée de vie.

L'agent de répertoire peut être découvert de trois façons différentes. Dans le premier cas la découverte est basée sur la diffusion de message et est soit active ou passive. Dans le cas actif, les agents d'utilisateur et les agents de service diffusent des requêtes auxquelles répond l'agent de répertoire. Dans le cas passif, l'agent de répertoire diffuse des annonces de sa présence. L'agent de répertoire peut également être découvert par un fichier de configuration ou par une option du protocole DHCP.

Lorsqu'il n'y a pas d'agent de répertoire, les agents de service répondent directement aux requêtes diffusées par les agents d'utilisateur.

3. Les composants répartis et CORBA

On peut voir la notion de composant réparti comme étant l'application du modèle *Remote Procedure Call* aux langages orientés objet. Les composants répartis apportent la transparence de localisation. Des objets se trouvant sur d'autres machines sont utilisés de la même manière que s'ils étaient sur la machine locale. Il se dégage actuellement deux architectures de composants répartis, le *Common Object Request Broker Architecture* (CORBA) et le *Distributed Component Object Model* (DCOM). Ces architectures sont appelées intergiciels (*middleware*) de composants répartis parce qu'elles font le lien entre les composants qui résident sur différentes plates-formes et qui coopèrent ensemble comme s'ils étaient tous sur une même machine. Les programmeurs utilisent les composants répartis de manière uniforme indépendamment des types de plate-forme, des langages de programmation et des protocoles de communication en cause. Pour y parvenir, elles automatisent certaines tâches de la programmation réseau telles que la découverte et l'activation des composants, l'encodage et le décodage des paramètres et le traitement des erreurs. Un composant est un objet local ou distant.

Un composant peut à la fois être client et serveur selon qu'il envoie ou qu'il reçoive une requête. Un composant client accède à un composant serveur par l'entremise d'une référence et d'une interface. La référence évite au composant client d'avoir à connaître la localisation du composant serveur. L'interface sépare le client de l'implantation du serveur. La localisation et l'implantation des composants peuvent changer sans avoir à modifier les applications qui les utilisent.

CORBA est une architecture de composants répartis dont la normalisation est effectuée par l'*Object Management Group* (OMG) [OMG 00]. L'architecture de gestion d'objets de CORBA, appelée *Object Management Architecture* (OMA), définit des règles pour que l'interaction entre les composants soit indépendante des protocoles de réseaux utilisés. Cette architecture est composée d'un modèle objet et d'un modèle de référence [OMG 97]. Le modèle objet définit la sémantique des caractéristiques des composants indépendamment de l'implantation choisie. La spécification des caractéristiques des composants est effectuée avec un *Interface Definition Language* (IDL) et leur implantation dans un langage de programmation

donné. Dans le modèle de l'OMA, un composant a une identité distincte et ses services peuvent être accédés à partir d'interfaces bien définies.

Un *Object Request Broker* (ORB) permet aux composants de communiquer dans un environnement réparti de façon transparente. Il permet à des programmes d'appeler des méthodes sur des composants tout en faisant abstraction de leur position géographique, du langage de programmation dans lequel ils sont implantés, du système sur lesquels ils se trouvent et des protocoles de communication. Sa tâche consiste à fournir des services pour localiser un composant serveur qui implante un service indiqué dans une requête d'un client, à établir une connexion avec le composant serveur, lui communiquer les données de la requête, à activer et désactiver les composants ainsi qu'à générer et interpréter les références aux composants.

Enfin, lorsqu'un programme client appelle une requête d'un composant serveur, le ORB du client et celui du serveur doivent nécessairement communiquer ensemble. Les protocoles inter-ORB sont utilisés pour la communication entre des ORB hétérogènes. Le *General Inter-ORB Protocol* (GIOP) est une spécification abstraite de la représentation des données et d'un ensemble de formats de messages qui peuvent être utilisés sur n'importe quel protocole de transport. Le *Internet Inter-ORB Protocol* (IIOP) spécifie comment GIOP peut être implanté sur le protocole de transport TCP.

Lorsqu'une requête est reçue par un ORB, celui-ci doit déterminer l'implantation du composant pouvant en prendre charge. Le ORB fait appel à un dispositif appelé l'adaptateur d'objet pour effectuer cette tâche. L'adaptateur d'objet permet l'enregistrement de l'implantation d'un composant dans un langage de programmation donné. Il est responsable de la génération d'une référence pour chaque composant CORBA, de trouver le composant serveur à qui s'adresse une requête et d'appeler la méthode appropriée sur ce composant. Si le composant en question n'est pas activé lors de l'arrivée de la requête, l'adaptateur d'objet se charge de le faire. C'est en particulier grâce à l'adaptateur d'objet que CORBA peut supporter diverses implantations de composant. CORBA permet d'avoir plusieurs adaptateurs d'objet qu'on appelle *Portable Object Adapter* (POA). On retrouve normalement un POA pour chaque langage de programmation.

4. Découverte de services à base de composants répartis

Cette section présente le système de localisation de services, à base de composants répartis, que nous avons réalisé.

Nous introduisons d'abord une application développée pour fins de démonstration. Il s'agit d'un bloc note électronique avec mécanisme d'impression (figure 2). Ce dernier est présenté à l'utilisateur au moyen d'une fenêtre dont fait partie une liste combinée énumérant les noms et les caractéristiques des imprimantes obtenues lors de processus de localisation de service antérieurs (figure 3).

Le contenu de la liste combinée peut être mis à jour, par le lancement d'un nouveau processus de localisation, mais au préalable l'utilisateur a le loisir de spécifier les propriétés d'imprimante désirées. Le bouton *Propriétés* (figure 3) ouvre une fenêtre de spécification (figure 4). Dans cet exemple, il en existe deux : le langage de l'imprimante et la taille du papier. L'utilisateur peut ne pas déterminer la valeur d'une propriété en sélectionnant *Quelconque* dans une liste de sélection. Un clic sur le bouton *OK* puis sur le bouton *Obtenir* déclenche le processus de localisation de service d'impression et entraîne la mise à jour de la liste combinée.

L'impression du contenu du bloc note est lancée par un clic sur le bouton *OK* de la fenêtre *Impression* (figure 3).

Figure 2. *Bloc note électronique.*

Figure 3. *Fenêtre d'impression.*

Figure 4. *Caractéristiques d'imprimante désirées.*

4.1. Architecture

Pour décrire l'architecture et le comportement de notre système, nous utilisons la notation *Unified Modeling Language* (UML) [RUM 98]. Une vue simplifiée de l'architecture est présentée à la figure 5. D'une part, il y a l'ANP sur lequel des applications fonctionnent. Celles-ci ont besoin de localiser et d'utiliser des services. D'autre part, il y a des éléments d'infrastructure CORBA, dont un serveur de noms, un service de localisation de ressources et des ressources matérielles et logicielles.

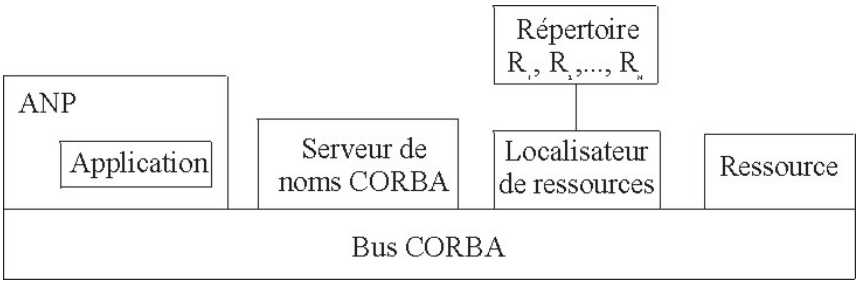

Figure 5. *Architecture simplifiée du logiciel.*

Au travers d'une application de gestion (non illustrée à la figure 5), l'administrateur du réseau crée, modifie et supprime des composants CORBA représentant des véritables ressources matérielles ou logicielles accessibles aux utilisateurs. Ces composants permettent aux utilisateurs de faire appel aux services des ressources qu'ils représentent. Au moment de leur création, les composants sont sérialisés et entreposés dans un répertoire (R_1, R_2,…,R_N).

Par la suite, l'administrateur du réseau lance un service de localisation de ressources : le *localisateur de ressources*. Au démarrage, ce dernier désérialise, charge en mémoire et inscrit dans son service les composants CORBA qui ont été créés et stockés dans le répertoire auparavant. Le localisateur de ressources s'enregistre auprès du serveur de noms CORBA. Il devient alors accessible aux applications utilisatrices de services. L'échange de requêtes et de réponses entre les applications et le localisateur de ressources se fait sur le bus CORBA par l'entremise d'un ORB présent sur la machine de chacun des partis.

Du côté ANP, une application requérant des services utilise le serveur de noms CORBA pour obtenir une référence au localisateur de ressources. Par la suite, elle emploie cette référence pour envoyer des requêtes au localisateur de ressources. Des références à des composants répartis représentant des ressources constituent les réponses des requêtes adressées au localisateur de ressources. La communication avec les ressources se fait au travers du bus CORBA. Ainsi, CORBA est employé à la fois pour découvrir et pour communiquer avec les ressources. Les interfaces de découverte et d'utilisation de services sont uniformes. Un seul protocole de communication est nécessaire pour soutenir toutes ces fonctions à l'intérieur de l'ANP.

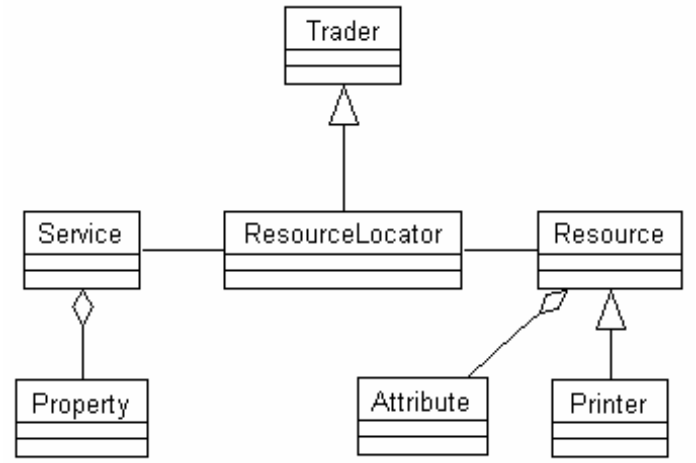

Figure 6. *Architecture détaillée du logiciel.*

L'architecture détaillée de la partie serveur est modélisée par le diagramme de classes de la figure 6. Chaque classe représente une définition de composants répartis CORBA. Les composants de la classe *Resource* sont des abstractions de ressources matérielles ou logicielles. Une ressource peut avoir plusieurs caractéristiques. Ces caractéristiques sont représentées par des composants de la classe *Attribute*. Une ressource peut posséder plusieurs *Attribute*. Les types de ressources sont créés sous forme de spécialisation de la classe *Resource*, ex. : la classe *Printer*.

Les composants de la classe *Service* représentent des requêtes formulées par des applications sur ANP. Une requête est un ensemble de propriétés (composants *Property*) qui décrivent un besoin.

ResourceLocator est la classe du composant de l'architecture qui fournit le service de localisation de ressources. Elle est une spécialisation de la classe *Trader* qui représente le service d'annuaire de type pages jaunes de CORBA. Le *Trader* permet d'enregistrer des composants ainsi que leurs propriétés, pour la recherche par propriétés. La classe *ResourceLocator* spécialise ce service en restreignant l'enregistrement de composants à ceux de type *Resource* seulement.

L'architecture de la partie localisation de services d'une application sur un ANP est présentée à la figure 7. Les classes *_st_ResourceLocator*, *_st_Resource* et *_st_Attribute* sont des souches (c'est-à-dire des classes C++) assurant la communication avec les composants CORBA de type *ResourceLocator*, *Resource* et *Attribute*. Les classes *ResourceSeq* et *AttributeSeq* permettent de manipuler des listes d'objets de type *_st_Resource* et *_st_Attribute*, respectivement.

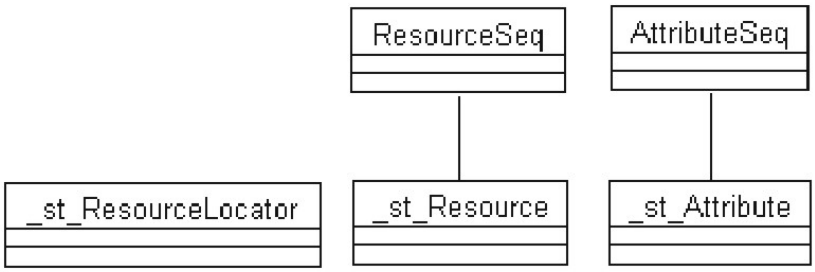

Figure 7. *Architecture de la partie localisation de services d'une application.*

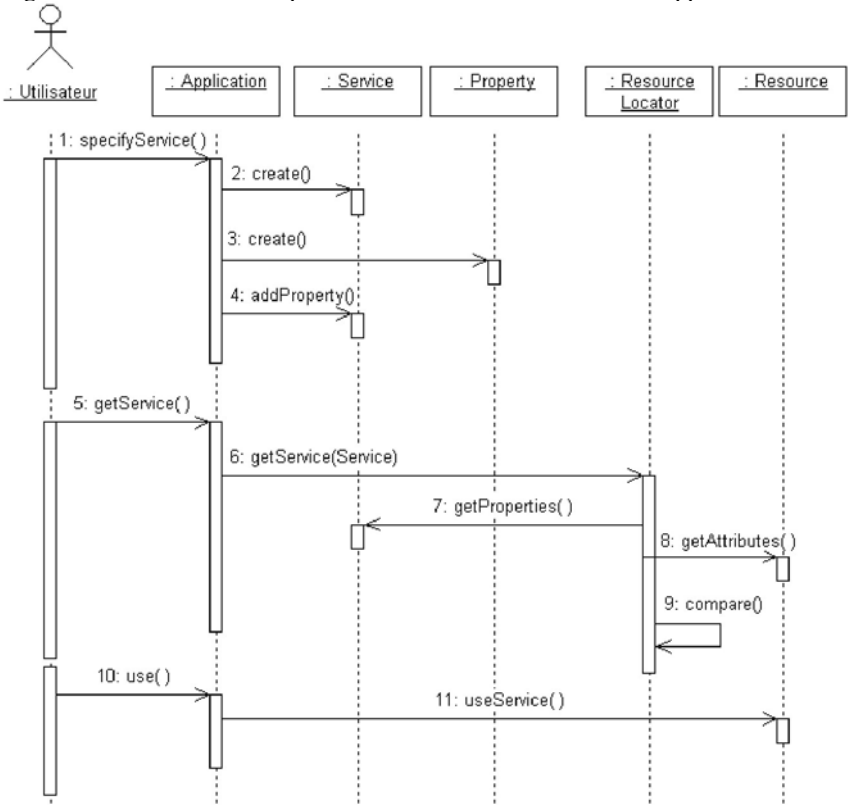

Figure 8. *Spécification, localisation et utilisation d'un service.*

Lorsqu'une application cherche des ressources qui fournissent les services dont elle a besoin, les requêtes sont passées au localisateur de ressources en utilisant une instance de la classe *_st_ResourceLocator*. Cette dernière retourne la liste (une instance de la classe *ResourceSeq*) des ressources retrouvées, des instances de la

classe *_st_Resource*. Pour utiliser une ressource, l'application adresse ses requêtes à l'instance de la classe *_st_Resource* qui la représente.

Un scénario de spécification, de localisation et d'utilisation d'un service est représenté par le diagramme de séquences de la figure 8. Pour simplifier la description, nous faisons abstraction de la mécanique des souches susmentionnée. La première séquence décrit la spécification d'un service par l'utilisateur au travers d'une application (1). L'application crée un composant de type *Service* (2) puis un ou plusieurs composants de type *Property* (3) afin de décrire les propriétés du service recherché. Les propriétés créées sont ajoutées du composant *Service* (4).

Le processus de localisation est déclenché par un appel à l'opération *getService()* (5). Le composant *Service* est transmis à un composant de type *ResourceLocator* (6). Les propriétés du composant de type *Service* (obtenus 7) et les attributs des composants de type *Resource* (obtenus en 8) sont comparés par *ResourceLocator* (9). *ResourceLocator* retourne la liste des composants de type *Resource* qui fournissent le service demandé, le cas échéant. L'utilisation d'un composant de la classe *Resource* au travers d'une application est modélisée par la troisième séquence : appel de l'opération *use()* sur l'application (10) puis de l'opération *useService()* sur un composant de type *Resource* (11). Toutes ces interactions se font par l'entremise du bus CORBA.

Pour que l'application puisse s'adresser au localisateur de ressources, elle doit connaître l'adresse IP du nœud du réseau qui exécute le serveur de noms CORBA. L'utilisateur spécifie un ensemble de telles adresses à partir d'une application. Elles sont enregistrées dans une base de données. Le contenu de cette base de données est éventuellement présenté à l'utilisateur pour qu'il puisse choisir l'adresse IP d'un serveur de noms CORBA lorsqu'il est connecté à un réseau. Par la suite, l'adresse est fournie au composant *_st_ResourceLocator* au moment de sa création.

Le langage de programmation Java a été utilisé pour implanter le localisateur de ressources, ce qui le rend exécutable sur plusieurs plates-formes. Le ORB VisiBroker [BOR 00] pour Java de Inprise a été utilisé pour le développement des composants répartis.

Les applications s'exécutant sur un ANP Palm Pilot de 3Com sont programmées en langage C++. L'implantation du protocole IIOP de CORBA pour Palm Pilot, Mico, également développé en C++ par l'Université de Frankfurt, est utilisé pour la communication avec les composants répartis. L'API Win32 de Windows 95 a été utilisée pour implanter l'envoi de fichier à l'imprimante.

5. Discussion

CORBA facilite le développement des systèmes répartis. L'indépendance qu'il permet face à la localisation des composants permet au programmeur de ne pas se soucier des détails associés aux communications distantes. En général, un coût est associé à l'emploi de fonctionnalités avancées comme celles fournies par CORBA. Ainsi, la transparence de localisation qu'offre CORBA peut nous amener à nous

demander si la prise en charge de cette fonctionnalité par le ORB occasionne des coûts supplémentaires, entre autres en ce qui concerne le trafic généré sur le réseau. Afin de nous éclairer sur ce point, nous avons inspecté le trafic engendré par l'utilisation de notre logiciel.

Puisque nous utilisons le protocole IIOP pour la communication entre les ORB, les messages échangés sont transportés dans des segments du protocole TCP. L'analyse du trafic capturé pendant l'utilisation de notre logiciel révèle que le nombre de segments échangés par les ORB et leurs composants est comparable au nombre de segments qui seraient échangés en utilisant un modèle client-serveur.

À la création d'un composant, les segments *Synchronise*, *Ack Synchronise* et *Ack* sont échangés entre le créateur du composant (une référence d'objet) et le POA désiré. Ces trois segments servent à établir une connexion TCP. Lors de l'appel d'une méthode du composant, la référence d'objet envoi le segment *Ack Push* est envoyé au POA du composant. Ce segment contient le nom du composant, le nom de la méthode ainsi que les valeurs des paramètres de la méthode. Si la méthode retourne une donnée, le segment *Ack Push* est envoyé par le POA, alors que le segement *Ack* est envoyé si la méthode ne retourne pas une donnée. La référence d'objet envoie ensuite le segment *Ack* pour accuser réception de la réponse du POA le cas échéant. Enfin, lorsque la référence est détruite, quatre segments sont échangés. Ces segments servent à effectuer la procédure normale de fermeture d'une connexion TCP.

En plus de la quantité de segments échangés, la taille de ceux-ci peut affecter l'utilisation de la largeur de bande d'un réseau. En effet, plus la taille des messages transmis par une application est grande, plus la largeur de bande est monopolisée par cette application. Nous avons examiné les frais fixes au niveau des messages du protocole IIOP. Les messages IIOP sont échangés seulement lors des appels de méthodes et des réponses à ces appels. Lors de l'appel d'une méthode, l'en-tête du message IIOP contient toujours 44 octets. Le reste du message est de longueur variable. Il contient le type et le nom unique du composant sur lequel la méthode est appelée, le nom de la méthode et les paramètres de la méthode. Pour une réponse, l'en-tête du message est de 28 octets. Le reste du message contient les données retournées par la méthode. Comme on est en mesure de le constater, les frais fixes du protocole IIOP pour la taille des messages sont minimes et n'impliquent pas un usage important de la largeur de bande.

Après cette vérification, nous pouvons affirmer que l'utilisation de CORBA, en ce qui concerne le protocole IIOP, a un impact très négligeable sur le trafic du réseau comparativement à l'avantage qu'il apporte. En ce qui concerne l'usage des composants répartis, il est certain que si un appel individuel de méthodes est effectué sur un composant pour obtenir chaque élément de données, le nombre de messages échangés sur le réseau sera grand. Il est de la responsabilité du développeur de prévoir le transfert en lot de données afin de diminuer le trafic sur le réseau.

Des améliorations peuvent être apportées à notre logiciel. Tout d'abord, l'obtention de l'adresse IP du serveur de localisation de ressources pourrait se faire

automatiquement. En effet, cette adresse peut être obtenue par la diffusion d'un message sur le réseau plutôt que d'exiger la saisie de cette donnée par l'utilisateur au travers de l'application de localisation de services.

Actuellement, les applications sur ANP sont de taille relativement grande (entre 59 et 66 kilo octets), versus 4 à 10 kilo octets pour les applications standards sur Palm OS. Ceci est dû en partie à l'ensemble du code des souches des composants répartis. Toutes les applications qui localisent et utilisent des services ont le code objet de ces fonctionnalités intégré dans leur programme exécutable. Le moyen de diminuer la taille des applications est de mettre le code partagé par toutes les applications dans une bibliothèque de liens dynamiques. Malheureusement, ce type d'unité logicielle n'existe pas sur le système d'exploitation Palm OS. S'il le devenait, cette solution serait à envisager afin de réduire la taille des applications.

Au niveau de la gestion des ressources, un certain nombre de propriétés pourraient être découvertes automatiquement lorsque l'administrateur définit les ressources. Pour la plupart des ressources d'un réseau, il existe des MIB SNMP d'où pourrait être extraites les caractéristiques des ressources.

Des mécanismes d'authentification devraient être implantés pour déterminer les droits d'accès au localisateur de ressources ainsi que les droits d'utilisation de chacune des ressources.

6. Conclusion

Le nombre d'utilisateurs d'ordinateurs et d'appareils de communication mobiles est en croissance constante. Ces appareils sont de plus en plus petits afin de faciliter leur transport et l'utilisation au cours des activités des utilisateurs. La petite taille de ces dispositifs implique une diminution des capacités des ressources matérielles (ex. : la mémoire, le processeur) qui les composent. Il n'en demeure pas moins qu'un appareil, tel un assistant numérique personnel, puisse faire partie d'un réseau informatique. Cela implique que tous les services du réseau soient mis à sa disposition de la même façon que pour les ordinateurs de taille plus grande. De plus, puisqu'il est mobile, il est probable qu'il doive utiliser les services de plusieurs réseaux différents. Cette mobilité oblige les utilisateurs à configurer leurs applications pour utiliser les services d'un réseau donné à chaque fois qu'ils changent d'endroit géographique. Pour ce faire, ils doivent découvrir les ressources du réseau qui offrent les services dont ils ont besoin. Plusieurs protocoles existent présentement pour la localisation de services sur un réseau.

La contribution apportée par la recherche présentée dans cet article est un système de gestion et localisation de services sur un assistant numérique personnel. La plupart des protocoles qui existent présentement permettent seulement de localiser les services. L'utilisation des services dans les applications nécessite une connaissance des protocoles utilisés par les services et parfois même de l'implantation des services. Les protocoles qui permettent un accès direct aux services ne sont pratiquement pas utilisables présentement sur les assistants numériques personnels à cause des technologies qu'ils utilisent. Notre logiciel, en

plus de permettre la localisation de services, permet également l'utilisation des services directement à partir des références sur les ressources offrant ces services. L'utilisation de CORBA et des composants répartis apporte une indépendance aux applications face à l'implantation des services, de même qu'une facilité d'utilisation.

De plus en plus, des entreprises de différents secteurs reçoivent pour de courtes périodes des travailleurs externes qui viennent leurs apporter expertises ou services. Beaucoup de ces travailleurs transportent avec eux leur petit ordinateur mobile afin d'effectuer le travail. Il en est de même pour les personnes qui sont amenées à voyager souvent pour le travail. Ces personnes peuvent avoir besoin d'utiliser des ressources du réseau où ils se trouvent pour obtenir des services tels l'impression d'un document, l'envoi d'un fax, Internet, et tout cela à partir de leur ordinateur mobile. La présence d'un logiciel de gestion et de localisation de services sur le réseau devient donc presque une nécessité pour faciliter la découverte et l'utilisation des services.

Bibliographie

[ALE 97] Alexander S., Droms R., « DHCP options BOOTP vendor extensions », Request For Comment 2132, Internet Engineering Task Force, 1997.

[BOR 00] Borland Inprise, VisiBroker, http://www.inprise.com/visibroker.

[CON 98] Conta A., Deering S., Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification, Request for Comments 2463, Internet Engineering Task Force, 1998.

[DEE 98] Deering S., Hinden R., Internet Protocol, Version 6 (IPv6) Specification, Request for Comments 2460, Internet Engineering Task Force, 1998.

[DEE 91] Deering S. (Editor), « ICMP Router Discovery Messages, » Request for Comments 1256, Internet Engineering Task Force, 1991.

[DRO 97] Droms R., « Dynamic Host Configuration Protocol, » Request For Comment 2131, Internet Engineering Task Force, 1997.

[KEI 99] Edwards W.K., *Core Jini*, Prentice Hall PTR, 1999.

[KEM 99] Kempf J., St. Pierre P., *Service Location Protocol for enterprise networks*, John Wiley & Sons, Inc., 1999.

[LEA 97] Leach P., Naik D., « CIFS/E Browser Protocol : Preliminary Draft », Informational Draft, Network Workging Group, Hanuary 1997.

[NAR 98] Narten T., Nordmark E., Simpson W., Neighbor Discovery for IP Version 6 (IPv6), Request for Comments 2461, Internet Engineering Task Force, 1998.

[OMG 00] http://www.omg.org.

[OMG 97] Object Managament Group, « A Discussion of the Object Management Architecture, White Paper, 1997.

[PER 96a] Perkins C., Harjono H., « Resource Discovery Protocol for Mobile Computing, » *Mobile Comunications, Technology, Tools, Applications, Authentication and Security*, IFIP World Conference on Mobile, Communications, Canberra, Australia, Chapman & Hall, London, UK, 1996, pp. 219-236.

[PER 96b] Perkins C. (Editor), IP Mobility Support, Request for Comments 2002, Internet Engineering Task Force, 1996.

[RUM 98] Rumbaugh J., Jacobson I., Booch G., *The Unified Modeling Language Reference Manual*, Addison-Wesley, 1998.

[SAN 95] Sant'Angelo R., *NetWare Unleashed*, Indianapolis, IN, Sams publishing, 1995.

[SID 90] Sidhu G., Andrews R., Oppenheimer, R. *Inside Appletalk : Second edition*, Reading, MA, Addison-Wesley, 1990.

[SUN 99] Sun Microsystems Inc., « Jini, » Architectural Overview, Technical White Paper, 1999.

[VEI 97] Veizedes J., Guttman E., Perkins C., Kaplan S., « Service Location Protocol, » Request For Comment 2165, Internet Engineering Task Force, 1997.

[WAH 97] Wahl M., Howes T., Kille S., « Lightweight Directory Access Protocol (v3), » Request For Comment 2251, Internet Engineering Task Force, 1997.

Biographies

François Lévesque a obtenu un baccalauréat en informatique, en 1996, et une maîtrise en informatique, en 2000, de l'Université de Sherbrooke.

Depuis novembre 1999, François Lévesque occupe le poste de chercheur en communications par satellites au Centre de recherche sur les communications à Ottawa. De février 1997 à juillet 1998 il a été analyste-programmeur dans une compagnie de développement de logiciels à Montréal.

Les domaines de recherche auxquels il s'intéresse particulièrement sont les réseaux de télécommunication, les télécommunications par satellites, les protocoles de télécommunication, les systèmes répartis et les systèmes temps réels.

Michel Barbeau détient un baccalauréat en informatique de l'Université de Sherbrooke (1985), une maîtrise en informatique de l'Université de Montréal (1987) et un doctorat en informatique de l'Université de Montréal (1991). Durant ses études de doctorat, il s'est mérité la médaille d'or académique du Gouverneur général du Canada.

Depuis janvier 2000, Michel Barbeau est professeur agrégé à Carleton University (Ottawa). Il a été professeur à l'Université de Sherbrooke de juin 1994 à décembre 1999. Il a été chercheur visiteur à University of Aizu (Japon) de juillet 1998 à juin 1999 et assistant de recherche à INRS-Télécommunications (Montréal) de mai 1987 à juillet 1988.

Ses principaux domaine de recherche sont les réseaux d'ordinateurs, les réseaux d'ordinateurs sans fils, les protocoles de télécommunications, les télécommunications par satellites, le génie logiciel et les systèmes temps réels.