Médiation Ambiante: sélection dynamique de ressources

Kim Tâm Huynh
Laboratoire PRiSM
Université de Versailles
Saint-Quentin-en-Yvelines
Versailles, France
kth@prism.uvsq.fr

Béatrice Finance
Laboratoire PRiSM
Université de Versailles
Saint-Quentin-en-Yvelines
Versailles, France
beatrice@prism.uvsq.fr

ABSTRACT

Nous adressons le problème d'intégration à la volée de nombreuses sources de données hétérogènes et autonomes. Notre but est de faciliter le développement d'applications sensibles au contexte embarquées sur des équipements mobiles. Dans ce but, nous avons proposé une nouvelle architecture de médiation de données ambiantes qui collecte des événements en captant l'environnement et les aggrège pour répondre aux besoins des applications. Dans ce papier, nous nous focalisons sur le problème de sélection de ressources dans un environnement dynamique, ce qui consiste en la correspondance entre les services requis par les applications et les services alentours fournis par les sources mobiles rencontrées par l'utilisateur. Le problème de sélection de ressources est continu et incrémental. Il prend en compte les connexions et déconnexions empiriques des ressources ambiantes. Son but est d'adapter continuellement l'ensemble des ressources connectées pour offrir une continuité de services. Dans ce papier, le problème de sélection de ressources dans un environnement dynamique est formalisé.

1. INTRODUCTION

Dans le domaine des réseaux de capteurs, la collecte de données est une tâche fondamentale. Les capteurs génèrent des données et les transmettent généralement vers un ou plusieurs nœuds serveur pour une analyse et un traitement ultérieur plus poussé. Les capteurs et les nœuds serveur sont vus comme un ensemble de ressources ambiantes qui peuvent être fixes ou mobiles. La consommation d'énergie de ces équipements et leur éloignement peuvent affecter l'utilisation de ces ressources et dégrader la qualité de services. Plusieurs architectures et approches ont été proposées pour traiter ce problème [7].

Dans ce papier, nous considérons une certaine infrastructure de médiation de données où les ressources ambiantes sont fixes ou mobiles et où un seul nœud serveur, appelé système de médiation, est considéré. Ce médiateur peut lui aussi être fixe ou mobile. En effet, aujourd'hui de plus en plus d'applications ambiantes sont développées sur des équipements mobiles tels que les smart phones et l'objectif du système de médiation de données ambiant est de faciliter le développement de ces applications souvent contextuelles et personnalisées, et qui ont besoin d'interagir avec leur environnement, et en particulier avec des sources de données hétérogènes et autonomes, faiblement couplées et distribuées.

La principale contrainte d'un tel système de médiation de données est la mobilité à la fois des ressources et du médiateur, ce qui rend incontrôlable les connexions et déconnexions de ressources quelque soit la raison (c.a.d., manque d'énergie, panne, éloignement). Nos hypothèses se basent sur une informatique dite opportuniste "qui exploite la mobilité des humains et leur nature grégaire à permettre une transmission seulement si deux équipements sont suffisamment proches" [2]. L'informatique opportuniste exploite les ressources disponibles dans un environnement donné pour fournir des services de calcul collaboratif aux applications et utilisateurs.

Dans cet article, nous nous intéressons surtout au problème de la sélection de ressources dans ce type d'environnement dynamique. Cela consiste à sélectionner les ressources de l'environnement qui fournissent des services requis par les applications déployées au dessus du système de médiation. Les ressources et l'utilisateur étant mobiles, l'environnement ambiant change constamment obligeant le processus de sélection de ressources à s'adapter en continu au nouveau contexte. Il permet au système de médiation de se connecter dynamiquement aux ressources sélectionnées aussi longtemps qu'elles restent actives (i.e., visibles) pour récupérer les données.

Dans ce contexte, le système de médiation de données ambiantes offre une vue unifiée des ressources disponibles en cachant leurs connexions et déconnexions, et en fournissant tant que possible une continuité de services aux applications des utilisateurs. Pour minimiser la consommation d'énergie, nous faisons l'hypothèse d'une communication directe entre le médiateur et n'importe quel ressource disponible. Évidemment, ayant imposé ces contraintes, nous ne prétendons pas proposer une solution générique pour la collecte de données dans les réseaux de capteurs mais nous essayons plutôt de fournir une approche originale et efficace à une certaine classe d'applications d'intelligence ambiante dite "in the small", qui est plus centrée sur l'utilisateur qui évolue dans un envi-

ronnement physique proche et qui a besoin de tirer parti au mieux de la présence de dispositifs et objets communicants pour faciliter son activité.

Pour donner une idée du problème adressé dans ce papier, nous considérons l'exemple suivant. Supposons que nous avons une classe d'applications intéressée par cet ensemble de services: $S_r = \{s_1, s_2, s_3, s_4\}$. Supposons qu'il existe un ensemble de ressources $\mathcal{R} = \{r_1, ..., r_n\}$, chacune fournissant un ou plusieurs des précédents services. A un instant donné t, seul un sous-ensemble $\mathcal{R}_a$ de $\mathcal{R}$ est actif, avec la contrainte que chaque ressource r_i de $\mathcal{R}_a$ a la probabilité $Pr(r_i, \Delta t)$ de se déconnecter durant l'intervalle Δt. Le problème consiste à: (i) trouver les ensembles de ressources qui fournissent les services requis, (ii) ordonner ces ensembles de ressources selon la meilleure satisfaction des besoins de l'application, tout en minimisant les probabilités de déconnexion des ressources. De plus, le fait que le processus soit continu ajoute une complexité majeure au problème.

Le papier sera organisé de cette façon. En Section 2, l'architecture avec son environnement ambiant, ses applications, et le système de médiation est brièvement présentée. En Section 3, le problème de sélection de ressources dans un environnement dynamique est posé formellement. La Section 4 compare notre approche aux travaux existants. Enfin, nous terminerons par une conclusion dans la Section 5.

2. ARCHITECTURE DE MÉDIATION DE DONNÉES AMBIANTES

Le système de médiation joue le rôle d'un nœud serveur mobile. Il est embarqué dans un équipement mobile tel qu'un smart phone, une tablette ou même un ordinateur portable. Il permet à une classe d'applications de souscrire à un ensemble de services à travers un ensemble de règles actives contextuelles et personnalisées [6]. Cette section donne un aperçu des ressources ambiantes, la définition d'un environnement ambiant, la description du besoin des applications, et une idée de l'architecture du médiateur ambiant.

Afin de faire inter-opérer les applications ambiantes avec les ressources (ou dispositifs) ambiants via le système de médiation, nous nous basons sur une architecture faiblement couplée et faisons l'hypothèse dans cet article qu'un schéma ou ontologie global a été défini par un groupe de standardisation; que les ressources implémentent ces services normalisés; et que les applications font référence à ces mêmes services.

2.1 Environnement ambiant

Une *ressource ambiante* est un capteur, un actionneur ou n'importe quel système ou équipement autonome proposant un service fournissant des données ou contrôlant l'environnement. Nous supposons que nous avons la connaissance de toutes les ressources existantes composant $\mathcal{R}$ et de tous les services $\mathcal{S}$ offerts par $\mathcal{R}$. Chaque ressource $r \in \mathcal{R}$ est décrit par un triplet de méta-données $< Id_r, D_r, S_r >$; où Id_r est l'identifiant de la ressource, D_r est une liste d'attributs décrivant r, et S_r l'ensemble des services offerts par r ($S_r = \{s \in \mathcal{S} | s$ est offert par $r\}$).

Chaque *service* s fourni par une ressource ambiante r est défini par sa signature: $s(X) \rightarrow Y$ où X et Y représentent respectivement les paramètres d'entrées et de sortie du service. Chaque élément de X ou de Y est typé: $< varName : varType >$.

Une *ressource active* est une ressource qui est visible, à un instant donné et durant un intervalle de temps, à travers le réseau, et prêt à fournir un service donné. Un service d'une ressource active peut délivrer un flux de données qui est d'un certain intérêt pour les applications de l'utilisateur. La consommation d'énergie d'une ressource active ou son éloignement par rapport au médiateur en raison de la mobilité du médiateur ou de la ressource, peut entraîner une déconnexion de la ressource ou bien la rendre inutile. Nous supposons que l'ensemble des ressources actives nous est fourni au médiateur à une certaine fréquence.

Un *état ambiant* (R_a, l, t) est défini par l'ensemble des ressources actives R_a, à un certain contexte de l'utilisateur défini par le lieu l et l'instant t.

Un *environnement ambiant Env* est défini comme un flux d'états ambiants, qui est potentiellement une séquence infinie d'états (R_a, l, t).

Le système de médiation est défini pour être à l'écoute de l'environnement ambiant, collecter et agréger les données fournies pour satisfaire les besoins des applications.

2.2 Besoins des applications

Les besoins des applications sont définis comme un ensemble de services requis par les applications: $Q = \{s_1, ..., s_n\}$ avec $s_i \in \mathcal{S}$.

En réalité, les applications souscrivent au système de médiation au travers d'un ensemble d'événements typés et de règles actives de type ECA (Evénement-Condition-Action). Cette section n'a pas pour vocation détailler la manière dont les événements sont générés, ni de dire comment les règles sont évaluées, mais plutôt de donner une intuition sur la façon dont Q est dérivé à partir de la spécification des événements et des règles. Pour cela, nous considérons l'exemple de deux applications, l'une permettant d'allumer le chauffage d'une pièce si la température détectée dans la pièce est inférieure à un certain seuil et l'autre déclenchant une alarme si une personne non autorisée est détectée.

Considérons l'ensemble des services suivants dont certains ne nécessitent aucun paramètre d'entrée ou de sortie:
`-monitorTemperature()` → `[temp:{val:real, id_sensor:real}]` retourne un flux de données de température qui contient sa valeur et l'identifiant du capteur.
`-getLocation(id_sensor:real)` → `location:{val:string}` retourne un tuple représentant le lieu dans lequel se trouve le capteur.
`-turnHeat(location:string)` permet d'allumer le chauffage dans le lieu donné en paramètre d'entrée.
`-monitorPresence()` → `[presence:{id_person:real, id_detector:real}]` retourne un flux de données de présences détectées qui contient l'identifiant de la personne détectée et l'identifiant du détecteur.
`-getPerson(id_person:real)` → `person:{name:string}` retourne un tuple représentant le nom de la personne détectée

à partir de son identifiant.
-`getRoom(id_detector:real)` → `room:{val:string}` retourne un tuple qui contient le nom de la pièce où le détecteur se situe.
-`alarm()` déclenche une alarme d'intrusion.

Soient les objets persistants du système de médiation:
- `Auth_People:[name:string]` est une relation contenant les noms des personnes autorisées.
- `Context_location:string` est un attribut représentant la localisation du système de médiation.

En premier lieu, le système de médiation a besoin de connaître quels sont les événements pertinents pour les applications. Par conséquent, les applications ambiantes doivent déclarer les types d'événements à considérer en donnant leur schéma et la requête utilisée pour générer l'événement. Dans ce papier, nous utilisons un modèle simplifié qui consiste à donner le schéma de l'événement et l'ensemble des services requis pour générer une instance d'événement. Les requêtes qui impliquent les services sont omises ici. Noter que les les services sont vus comme des relations virtuelles. Intuitivement, le système de médiation va exécuter la requête dans laquelle les services sont impliqués, agréger les données pour finalement instancier l'événement.

Dans cet exemple, les applications ambiantes sont intéressées par les événements suivants:
- `Temperature`:{value:real, timestamp:real, location:string}, $\mathcal{S}_{Temperature}$ = {monitorTemperature, getLocation}
- `PersonDetected`: {name:string, timestamp:real, location: string}, $\mathcal{S}_{PersonDetected}$ = {monitorPresence, getPerson, getRoom}
Par exemple, pour le second événement, le système de médiation va invoquer `monitorPresence` pour contrôler la présence d'individus, `getPerson` pour connaître qui a été détecté et `getRoom` pour connaître le lieu où la personne a été détectée. Puis, il va agréger ces deux informations pour produire un événement `PersonDetected` sur la base de la requête définie pour cet événement.

Ensuite, les applications ambiantes spécifient leurs règles ECA. Par exemple, la première règle allume le chauffage si un événement `Temperature` est détecté dans le même lieu que l'utilisateur et que sa valeur est inférieure à un certain seuil.

on Temperature t
if t.value < 15 and t.location = Context_location
then turnHeat(t.location)

La seconde règle déclenche une alarme si l'événement `PersonDetected` détecté concerne une personne non autorisée à se trouver dans ce lieu.

on PersonDetected p
if p.name not in Auth_People.name
then alarm()

Les besoins des applications, dénoté Q, sont dérivés à partir de la spécification des événements et des règles ECA. Q contient l'ensemble de tous les services utilisés pour instancier les événements, évaluer les conditions ou exécuter les actions des règles. Dans cet exemple, Q est égal à { monitorTemperature, getLocation, turnHeat, monitorPresence, getPerson, getRoom, alarm}.

2.3 Système de médiation

Le système de médiation relie de façon opportuniste les besoins de l'application Q avec l'environnement ambiant grâce à différents composants illustrés dans la Figure 1.

Le sélecteur de ressources choisit un ensemble de ressources actives fournissant les services requis par les applications. Les collecteurs de données récupèrent les données en invoquant des services spécifiques fournis par le sélecteur de données. Le composeur d'événements utilisent les événements fournis par les collecteurs de données pour inférer des événements plus complexes qui peuvent déclencher les règles actives. L'évaluateur de règles exécute les règles actives des applications. Pour plus de détails, voir les travaux [6]. Dans ce papier, nous nous focalisons uniquement sur une tâche spécifique du médiateur: 'le sélecteur de ressources'.

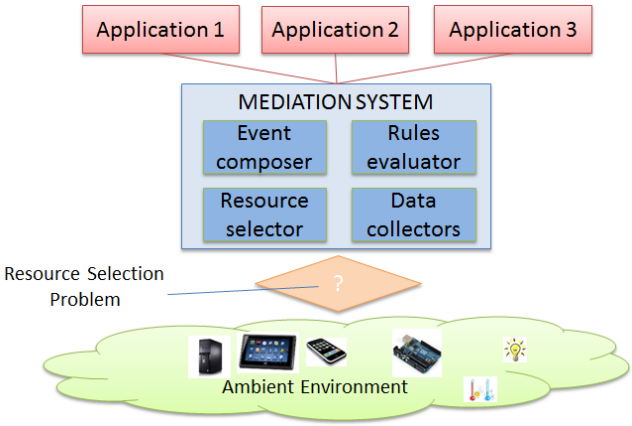

Figure 1: Système de médiation de données

L'extraction de Q à partir des règles actives est assez simple, cependant l'évaluation de Q pose plusieurs problèmes. En effet, le sélecteur de ressources a besoin de réévaluer Q dès que l'environnement change. Cependant cela peut être trop coûteux de le faire à chaque changement de contexte car si on le fait trop souvent on n'a pas le temps de s'adapter qu'un autre contexte a déjà changé. Ici on préfèrera plutôt le faire à une certaine fréquence afin de garantir une continuité de services, notamment pour les services d'acquisition de données pour le bon fonctionnement des applications.

Par ailleurs, comme on ne dispose pas forcément de tous les services requis dans l'environnement ambiant au même instant, il s'agit de trouver les ressources qui satisfont au mieux les besoins des applications; et comme Q est évaluée continuellement, le problème principal est de fournir efficacement une solution dans un temps raisonnable. Dans la prochaine section, nous formalisons cette problématique.

3. PROBLÉMATIQUE

Le problème de la sélection de ressources consiste à identifier à partir des ressources actives celles qui satisfont les besoins de l'application. La sélection est faite à la volée tant que le flux de ressources décrivant l'environnement varie d'un état

à un autre. La sélection de ressource est la première phase avant l'acquisition des données et l'exécution des règles.

3.1 Evaluation de Q

Étant donné un environnement ambiant Env défini comme un flux d'éléments (R_a, l, t) où R_a est l'ensemble des ressource actives dans un lieu l à un instant t et le besoin Q défini comme un ensemble d'éléments s_i où s_i est un service requis; satisfaire le besoin Q à partir de Env consiste à sélectionner continuellement les ressources pertinentes qui sont susceptibles de *couvrir* au mieux Q et qui ont la plus faible *probabilité* de se déconnecter.

Plus précisément, soit $r.s$ une notation d'un service s fourni par la ressource $r \in R_a$, le problème de la sélection de ressources consiste à calculer continuellement et successivement les ensembles suivants:

1. $RR_a \subseteq R_a$ est l'ensemble des ressources pertinentes, défini comme suit:
   $RR_a = \{r | (\exists r \in R_a, \exists s' \in Q | match(r.s, s') = 1)\}$
   où $match(r.s, s')$ est un 'matching' exact de chaines de caractères entre les signatures des services s et s'. $match$[1] est défini comme une fonction booléenne qui retourne 1 s'il y a correspondance entre les deux services et 0 sinon.

2. $\mathcal{P}(RR_a)$ est l'ensemble des parties (i.e., powerset) de RR_a, c'est le domaine sur lequel Q est interprété. Le problème ici est l'énumération des 2^p éléments où $p = |RR_a|$. Rappelons que les éléments du powerset sont des ensembles de ressources.

3. $I_Q = \{i_Q \in \mathcal{P}(RR_a) | i_Q \text{ est une solution de } Q\}$.
   $I_Q \subseteq \mathcal{P}(RR_a)$.
   i_Q est une solution de Q si:
   $\forall r \in i_Q, \nexists r' \in i_Q | (r' \neq r \text{ and } match(r'.s, r.s) = 1)$.
   Intuitivement, dans une solution, il ne peut y avoir qu'une seule ressource fournissant le même service.

La sélection des meilleures solutions $i_Q \in I_Q$ dépend de deux paramètres:

- la *couverture* du besoin Q, qui est le ratio entre le nombre de services requis fourni par la solution i_Q et le nombre total de services requis par Q.

- la *probabilité de déconnexion* de chaque ressource $r \in i_Q$. Plus cette probabilité est faible, plus r sera capable de délivrer des données plus longtemps. Cette probabilité peut être définie à partir du niveau de batterie de la ressource, ou à partir de la distance de la ressource par rapport au médiateur, ou à partir des deux. Nous supposerons que chaque ressource $r \in i_Q$ a une probabilité $Pr(r, \Delta t)$ de se déconnecter durant l'intervalle Δt qui nous est donnée. A partir de cette probabilité, nous pouvons calculer, pour une solution i_Q donnée, sa probabilité de devenir invalide ($Pr(i_Q, \Delta t) = \max_{r \in i_Q} Pr(r, \Delta t)$).

[1]D'autres techniques plus complexes pour le matching de services peuvent être utilisées, telles que celles utilisées dans les architectures à base de composants. Cependant cela ne remet pas en cause l'approche qui prend en compte la durée du processus de sélection de ressources complet et son adaptation dans un environnement plus ou moins dynamique.

Étant donné que le but est de couvrir au mieux l'ensemble des services requis de Q, la couverture est un critère prioritaire par rapport à la probabilité[2]. Cependant, si deux solutions ont la même couverture, la solution avec la plus faible probabilité est gardée. La meilleure solution n'étant pas unique, le sous-ensemble de solutions équivalentes est noté $I'_Q \subseteq I_Q$.

Comme mentionné avant, le problème de sélection de ressources est un processus continu qui opère sur un flux Env. Le processus doit donc évaluer les différents ensembles RR_a, I_Q et I'_Q à une certaine fréquence Δt et choisir seulement une solution parmi celles de I'_Q. Le calcul de la la meilleure solution parmi le powerset $\mathcal{P}(RR_a)$ doit se faire dans un temps raisonnable, c.a.d, inférieur à Δt. Δt représente le délai entre deux cycles d'exécution du processus de sélection.

Si le temps d'exécution du processus global est supérieur à Δt, il devient impossible de produire efficacement une solution i'_Q. Pour résoudre ce problème, il est important de contrôler et de borner la durée du processus. Des stratégies d'exécution sous contraintes doivent être définies pour toujours fournir une solution dans les temps même si ce n'est pas la meilleure. Évidemment cela peut impacter la couverture de la solution produite.

3.2 Adaptation

Entre deux cycles d'exécution, les ressources continuent à se connecter et se déconnecter. La déconnexion de ressources peut faire diminuer la couverture de la solution courante nommée i'_Q. Si trop de ressources se déconnectent, le besoin peut ne plus être couvert du tout.

Pour traiter ce problème, il est nécessaire d'introduire un processus d'adaptation qui détecte les changements dans l'environnement et propose une solution alternative si la solution courante est affectée par ces changements. Le but est de maintenir la couverture en substituant les ressources déconnectées par de nouvelles ou même d'augmenter la couverture si de nouvelles ressources deviennent disponibles. Le processus d'adaptation calcule une nouvelle solution à partir de la solution courante i'_Q.

On peut faire remarquer que la solution calculée ne fait pas forcément partie des meilleures solutions car le calcul est incrémental (c.a.d., les ressources toujours connectées sont gardées et donc leur probabilité de se déconnecter augmente).

[2]Notons que cette probabilité est calculée par le médiateur à partir des meta-données fournies par les ressources. Par exemple si une ressource est fortement mobile elle peut donner sa direction, sa vélocité ce qui nous permet d'estimer sa trajectoire dans le temps et de détecter si elle va sortir du champ de détection bluetooth par exemple durant l'intervalle de temps Δt. De façon générale, plusieurs métriques peuvent être définies pour estimer l'utilité d'une ressource au fil du temps

4. TRAVAUX CONNEXES

Généralement, la sélection de sources peut être classée en deux catégories: automatique ou manuelle [9]. En effet, un processus peut être chargé de sélectionner les sources pertinentes selon certains critères comme la qualité de la source ou la configuration du réseau. Cependant, quelquefois, l'utilisateur peut faire partie du processus en sélectionnant lui-même les sources qu'il considère comme pertinentes. Dans notre cas, la sélection est faite automatiquement parce que dans des environnements dynamiques, l'ensemble des sources change constamment dans le temps et cela peut être lourd pour l'utilisateur de les sélectionner manuellement.

La sélection automatique de sources consiste à choisir les sources qui répondent aux besoins des applications. Dans le domaine des bases de données; le focus est mis sur le matching entre les descriptions des sources et la requête à évaluer [4]. Dans le domaine des systèmes orientés services [3], la sélection de services est équivalente à un problème de composition dynamique de services. En effet, la composition est faite à l'exécution et le but est de trouver le meilleur ensemble de services qui répondent aux besoins des applications en définissant des algorithmes de composition utilisant la qualité de services (QoS) [10, 8]. Dans les deux cas, le processus s'effectue à chaque fois qu'un nouveau programme ou requête est défini. Cependant, rien n'a été proposé dans le contexte des requêtes continues. En effet, dans ces approches, une fois que les services sont choisis, ils ne changent pas durant l'exécution de la requête.

Dans notre problème, la sélection est faite à une certaine fréquence et doit prendre en compte les changements de l'environnement, aussi bien que la probabilité de déconnexion. Le processus de sélection doit fournir une continuité de service nécessaire à l'évaluation des règles actives dans un environnement dynamique. Si aucun service n'est disponible, le système de médiation ne peut collecter d'événements nouveaux, mais les règles continuent à s'exécuter et consomment les événements déjà collectés. Nous pouvons remarquer que beaucoup de travaux existent pour l'évaluation adaptative de requêtes [5, 1], notre approche est complémentaire à ces travaux. A notre connaissance, il n'existe pas de travaux similaires dans la littérature.

5. CONCLUSION

Dans ce papier court, nous avons formalisé le problème de la sélection continue de ressources dans un environnement dynamique. Les algorithmes sous-jacents ont été définis mais n'ont pas été présentés dans l'article.

Nous avons développé un premier prototype validant l'approche de médiation de données ambiantes sur téléphones ou tablettes Android. Un démonstrateur permet au médiateur d'acquérir des données ambiantes provenant de capteurs de type Arduino, de façon dynamique et à la volée, connectés en bluetooth. Grâce aux langages de requêtes déclaratifs et des règles ECA, il est possible de décrire une application ambiante, sensible au contexte et aux préférences de l'utilisateur, en quelques lignes.

Actuellement, nous travaillons sur la mise en place d'un modèle de simulation nous permettant d'avoir un environnement plus riche en objets communicants et d'analyser la qualité du processus de sélection de ressources en fonction de différentes configurations (i.e., taille de Q, de R, des taux de connexion et déconnexion, etc) que nous pouvons rencontrer. Cela permettra de mesurer les différentes stratégies d'adaptation et leur impact sur la couverture des besoins des applications au fil du temps.

6. REFERENCES

[1] R. Avnur and J. M. Hellerstein. Eddies: Continuously adaptive query processing. *SIGMOD Rec.*, 29(2):261–272, May 2000.

[2] M. Conti and M. Kumar. Opportunities in opportunistic computing. *Computer*, 43(1):42–50, 2010.

[3] S. Dustdar and W. Schreiner. A survey on web services composition. *Int. J. Web Grid Serv.*, 1(1):1–30, Aug. 2005.

[4] M. Grawunder. The dynaquest-framework for dynamic and adaptive source selection. *SIMULATION SERIES*, 35(1):221–226, 2003.

[5] J. M. Hellerstein, M. J. Franklin, S. Chandrasekaran, A. Deshpande, K. Hildrum, S. Madden, V. Raman, and M. A. Shah. Adaptive query processing: Technology in evolution. *IEEE DATA ENGINEERING BULLETIN*, 23:2000, 2000.

[6] K. T. Huynh, B. Finance, and M. Bouzeghoub. Towards an ambient data mediation system. In *Proceedings of the 2nd International Workshop on Information Management for Mobile Applications (IMMoA) in conjunction with VLDB*, Istanbul, Turkey, 2012.

[7] X. Li, A. Nayak, and I. Stojmenovic. *Sink Mobility in Wireless Sensor Networks*, pages 153–184. John Wiley & Sons, Inc., 2010.

[8] N. B. Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas, and V. Issarny. QoS-aware service composition in dynamic service oriented environments. In *Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '09, pages 7:1–7:20, New York, NY, USA, 2009. Springer-Verlag New York, Inc.

[9] C. N. Ververidis and G. C. Polyzos. Service discovery for mobile ad hoc networks: a survey of issues and techniques. *Commun. Surveys Tuts.*, 10(3):30–45, July 2008.

[10] T. Yu, Y. Zhang, and K.-J. Lin. Efficient algorithms for web services selection with end-to-end QoS constraints. *ACM Trans. Web*, 1(1), May 2007.