

Mémoire pair-à-pair pour l'informatique diffuse

Loïc SCHMIDT

Encadrant : Nathalie MITTON

Directeur : David SIMPLOT-RYL

INRIA Lille - Nord Europe

IRCICA/ LIFL, CNRS UMR 8022, Univ. Lille 1

Email : {loic.schmidt, nathalie.mitton, david.simplot}@inria.fr

Résumé—L'informatique diffuse est définie comme étant une « technologie invisible à des utilisateurs avec lesquels elle entretient des interactions permanentes ». Cette transparence de l'informatique est rendue possible grâce aux technologies sans contact d'objets mobiles tels que les RFID, les code-barres 1D, 2D, la NFC, etc. Une plateforme classique d'intergiciel pour ces technologies est constituée de lecteurs connectés à un serveur qui sert de lien entre ces lecteurs et les applications métiers. Une telle solution souffrira de problèmes de passage à l'échelle. Le pair-à-pair, principalement utilisé pour l'échange de fichiers, permet une répartition de stockage d'information de façon robuste et distribuée, mais n'est pas une solution optimale pour les particularités de l'informatique diffuse. En effet, les objets mobiles peuvent avoir à stocker en masse des informations emmagasinées, le réseau n'est pas constitué uniquement de pairs, etc. Une telle gestion de la mémoire pour l'informatique diffuse est importante pour la conception d'intergiciel RFID tels que ASPIRE.

I. INTRODUCTION

L'informatique diffuse est définie comme étant une « technologie invisible à des utilisateurs avec lesquels elle entretient des interactions permanentes » [6]. On peut citer en exemple un musée qui transmettra aux téléphones portables de ses visiteurs des informations complémentaires sur les oeuvres qu'ils sont en train de regarder. Ces objets mobiles (du code-barres 1D au PDA en passant par le RFID) communiquent avec un lecteur situé à proximité servant de passerelle avec le système. Celui-ci va générer des événements spécifiques à l'action à effectuer grâce aux informations provenant de l'objet mobile.

L'architecture classique consiste en un serveur qui sert de lien entre les applications métiers et les lecteurs. Bien qu'elle soit adaptée pour les petites structures, cette architecture souffrira de problèmes de passage à l'échelle. En effet, le serveur pourrait être rapidement saturé et une panne sur celui-ci paralyserait toute la structure.

Nous analysons dans la section II l'architecture classique et ses problèmes de passage à l'échelle. Puis nous examinons, dans la section III, le cas du pair-à-pair comme solution potentielle aux différents problèmes identifiés afin d'obtenir un système de mémoire réparti où les informations sont stockées de façon intelligente, dupliquées mais pas sur tous les serveurs, et faciles à récupérer. Enfin nous concluons dans la section IV en présentant des applications concrètes nécessitant un tel système.

II. PASSAGE À L'ÉCHELLE

Dans le contexte de l'informatique diffuse, les objets mobiles communiquent avec des applications. Cette communication est possible de deux façons : (i) intégrer ces interactions dans les applications métiers ; (ii) passer par un intergiciel permettant la circulation d'évènements entre les lecteurs et différentes applications métiers. Par exemple, lors du passage sous un porche lecteur de RFID, les informations contenues dans les RFID du chargement vont être lues par le lecteur et envoyées au système qui décidera alors de l'action à faire : il pourra enregistrer les informations d'heure et de lieu, interroger un éventuel capteur pour connaître la température actuelle de conservation des produits, demander à une application d'envoyer une alerte si cette température n'est pas adéquate, etc.

L'architecture de ce genre de middleware se présente sous la forme d'un serveur qui va servir de lien entre les applications métiers et les lecteurs (voir figure 1). Lors du passage à l'échelle, le serveur est de plus en plus sollicité, et une congestion peut très vite apparaître.

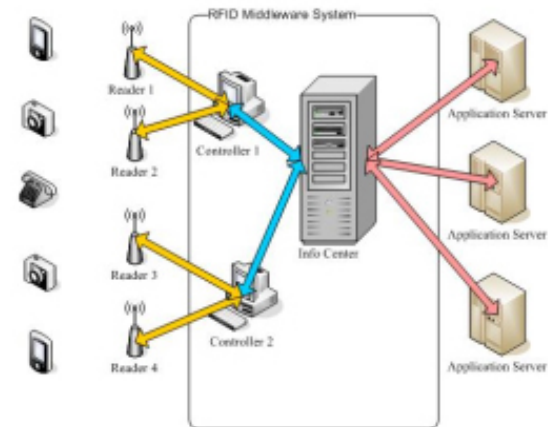


Figure: W/RFID Middleware Platform

Un moyen de parvenir à empêcher cette congestion serait de dupliquer les serveurs et les données. On aurait en plus résolu le problème de la tolérance aux pannes, mais on poserait alors le problème de l'intégrité de ces données. En effet, lors d'une modification d'une donnée, il faudra alors répercuter

cette modification sur chacun des serveurs, et vérifier qu'il n'y pas eu de lecture corrompue entre temps. De plus, la duplication des données impose un coût d'espace de stockage. Une autre solution consisterait à répartir les données sur plusieurs serveurs sans aucune redondance. Le problème de l'intégrité des données lors des écritures serait alors résolu ainsi que celui de l'espace de stockage, mais la robustesse serait compromise, et pour ce qui est de la congestion, elle peut être encore problématique si le système n'accède qu'à une seule et même donnée.

Nous nous intéressons à un compromis entre ces deux solutions qui, minimisant la redondance des données tout en assurant la tolérance aux pannes, offre une bonne répartition de ces données permettant une optimisation de l'utilisation de la bande passante.

III. SYSTÈME PAIR-À-PAIR (PÀP)

Dans cette section, nous présentons des systèmes de partage de fichiers en pair-à-pair existants, puis nous analysons les particularités d'un tel système dans le cas de l'informatique diffuse.

A. Solutions existantes pour le PàP

On remarque que dans les deux solutions présentées plus haut, le passage à l'échelle impose l'utilisation de plusieurs serveurs et une certaine redondance des données pour éviter la congestion et permettre une bonne tolérance aux pannes. Les systèmes connus de partage de fichiers en pair-à-pair (Gnutella [3], KaZaA [4], eDonkey [2], ...) permettent d'avoir des objets d'autant plus disponibles qu'ils sont populaires, et donc répliqués sur des nœuds. Cela permet alors de diminuer la charge imposée aux nœuds partageant les fichiers populaires. On voit que ce genre de système répond aussi bien à la problématique de surcharge du réseau qu'à celle de la tolérance aux pannes.

Le pair-à-pair offre aussi différentes techniques de localisation des données. En effet, il faut pouvoir récupérer les données réparties. Il existe plusieurs algorithmes dits de « lookup » [1]. Ces algorithmes utilisent des tables de hachages distribuées (DHT - Distributed Hash Table) permettant de localiser les données.

Mais le cas qui nous intéresse est particulier.

B. Particularités du PàP pour l'informatique diffuse

L'architecture de l'informatique diffuse est aussi composée de lecteurs mobiles pouvant lire et emmagasiner des données pour les transmettre ensuite en masse au système. Toute l'architecture du réseau n'est donc pas constituée de pair. Il faut aussi prendre en compte le traitement de ces données. En effet, les objectifs n'étant pas uniquement le stockage de données, mais aussi le traitement de celles-ci et des applications métiers. De plus, il y a les particularités connues a priori des types d'informations/traitements : base de données avec de petits contenus, traitements dédiés, etc.

Ces points nous montrent qu'il est nécessaire de proposer de nouvelles solutions visant à optimiser les axes présentés

ci dessus en s'inspirant des avantages du pair-à-pair et des caractéristiques de l'informatique diffuse afin de concevoir un système de mémoire réparti où les informations sont stockées de façon intelligente, dupliquées mais pas sur tous les serveurs, faciles à récupérer, tolérant aux pannes et passant à l'échelle.

IV. CONCLUSION

Les solutions pour l'informatique diffuse ne permettent pas le passage à l'échelle, dû au simple serveur faisant la passerelle entre les objets mobiles sans contacts et les applications métiers. Le pair-à-pair, distribuant les données de façon robuste, permet ce passage à l'échelle mais n'est pas optimal pour les particularités de l'informatique diffuse. En couplant les avantages du pair-à-pair à ces particularités, nous pouvons obtenir une gestion de la mémoire performante, robuste, et passant à l'échelle. Une telle gestion est utile dans des projets d'intergiciels RFID comme ASPIRE [5] ou plus général comme dans ICOM qui fait intervenir non seulement des RFID, mais tout types d'objets mobiles sans contact (NFC, code-barres 1D, 2D, RFID, etc.).

RÉFÉRENCES

- [1] Hari Balakrishnan, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Looking up data in p2p systems. *Commun. ACM*, 46(2) :43–48, 2003.
- [2] eDonkey protocol description. <http://kent.dl.sourceforge.net/pdonkey/eDonkey-protocol-0.6.2.html>.
- [3] Gnutella. <http://www.gnutella.com/>.
- [4] KaZaA. <http://www.kazaa.com/fr/index.htm>.
- [5] ASPIRE Advanced Sensors and lightweight Programmable middleware for Innovative Rfid Enterprise applications. <http://fp7-aspire.eu>.
- [6] Mark Weiser. Some computer science issues in ubiquitous computing. *Commun. ACM*, 36(7) :75–84, 1993.