Cet article propose de regrouper les ressources sous différents site et d'organiser de manière hiérarchique à l'intérieur d'un site. Le tout étant résistant aux connexions/déconnexions et passant à l'échelle.

Il est dans la thématique d'ubimob et la problématique est intéressante.

Le principal problème de cet article est d'identifier les contributions.

Les contributions sont :
Mise en place d'une spécification détachée du middleware P2P. Notre spécification est basée sur quelques propriétés de base : gestion de la connexité de la communauté, gestion du routage entre les nœuds. Nous n'exploitons pas plus de propriétés, et dans le cas de pastry, qui est notre exemple, nous n'utilisons pas les capacités des DHT car elles ne sont pas dans tous les middleware 2p
Mise en place d'une solution de stockage générique : l'utilisation des proxy data permettent de se détacher des types de stockage, lors de l'utilisation, un stockage sur disque local, serveur FTP ou en attachement de mail gmail sénat traite pour la lecture comme pour l'écriture par le proxy dat, et sera transparent pour l,utilisateur comme pour le système
L'accès aux données ne "souffre" pas des interconnexions possibles entre les noeud, en effet, la plupart des système P2P considère une recherche selon l'architecture et un échange direct entre les nœuds. Dans notre cas, si la connexion directe n,est pas possible, le transfert peut se faire par le biais du chemin de recherche de la ressource au sein de l'architecture hiérarchique

Je ne vois pas trop la différence avec l'article référencé [12]HP2P: A Hybrid Hierarchical P2P Network. J'ai l'impression que vous avez ajouté un index aux super-noeuds.

La notion de super nœud ne concerne pas vraiment notre architecture, en effet les nœuds ont une spécificité de rôle, mais toutes les spécificités peuvent être présente sur une même machine physique. L'architecture n'impose pas un rôle de super-nœud mais cela peut se faire dans les spécificités d'une implémentation.

Il faudrait expliquer comment cette structure fait diminuer le nombre de messages dans le cluster.

La recherche hiérarchique permet d'éviter l'inondation de tous les liens du réseau. La hiérarchisation permet de définir des chemins par lesquels transitent les requêtes, la connectivité logique de notre système est, par définition de (n-1) et permet donc de réduire par agrégation (algorithme de type PIF) le nombre de messages d'une recherche.

Des questions subsistent sur le système.

Qui place les noeuds de communication ?

Les nœuds de communication sont placés selon leur capacité et les besoins du système. Mais pour mémoire, un nœud physique peut supporter plusieurs plusieurs nœuds logiques, l'attribution du rôle de communicateur se fera en fonction de l'existence d'autres nœuds dans le réseau et aussi des besoins d'interconnexion avec d'autres communautés.

Comment communiquent-ils (topologie) ?
Comment se découvrent-ils ?

Les communications se font par la capacité de routage du middleware P2P croisé avec l'architecture de notre spécification. La topologie et l'implémentation des fonctions send/receive reste à charge du middleware utilisé.

Que se passe-t-il si C disparait, avons-nous une liste de C et de routage (via OSPF2) ?
Lorsqu'un nœud comme C disparaît, celui-ci peut être remplacé par un autre nœud choisi (un RM, par exemple). La reconnexion à d'autres communautés dépend toutefois d'autres facteurs comme la connaissance d'autres réseaux (par exemple, via un historique des requêtes déjà traitées), des politiques d'accès en place (SLAs, authentification) et aussi des capacités de communication (liaison directe, passage des pare-feu, etc).

Dans les détections de panne, vous détectez s'il est indirectement accessible. Comment utilisez-vous cette information ? Sachant que vous allez faire suivre au voisin proche la requête.
Pourquoi différente détection de pannes ? Qui envoie les battements de coeurs pour la détection de panne entre les couches ? (si c'est tout le monde, pourquoi s'embêter avec les jetons ?)

Ce mécanisme est suggéré comme une forme d'augmenter la fiabilité de la détection. D'autres formes de détection peuvent le remplacer, comme par exemple l'utilisation des mécanismes propres au réseau overlay.
L'usage des jetons a simplement l'avantage de permettre un diffusion uniforme de l'information : au retour du jeton, une machine est sûre d'avoir informé tous les autres nœuds. Si cette garantie n'est pas nécessaire ou si d'autres mécanismes plus performants sont disponibles, on peut remplacer l'usage des jetons.

À quoi sert l'élection d'un leader RM ? Sachant que dans vos exemples ils sont tous au même niveau ? Serait-ce une indexation supplémentaire ?

Comme maintenant indiqué dans le texte, l'élection d'un RM "leader" peut s'avérer nécessaire selon le type de distribution des données. Un cas typique serait un système de fichiers distribué comme HDFS, où un "NameNode" gère les index des différents "DataNodes".

Quand décide-t-on qu'on crée plus de RM ? uniquement qu'il n'y en a pas d'autre disponible ? (Élection avec un seul noeud ?)
Chaque DM doit être connecté à un RM dans son réseau ou, pour des questions de performance, dans la propre machine. Si aucun RM n'est disponible (ou si les spécificités du DM l'exigent), un RM peut être crée pour assurer cette tâche.

La liste des DM du RM et C ne doit-elle pas être connue par les DM pour effectuer une élection d'un nouveau RM parmi les DM ? Ce n'est pas précisé en cas de mort de tout les RM.
Dans le texte nous expliquons désormais plusieurs scénarios où l'élection peut avoir lieu. En général, il suffit d'avoir accès à l'historique des requêtes pour connaître un certain nombre de nœuds dans sa communauté (et dehors), grâce aux identifiants uniques C/RM/DM inclus dans chaque requête. Ainsi, si un nœud est promu, il peut puiser dans cet historique pour tisser ses connexions. D'autres mécanismes sont aussi possibles, comme par exemple l'appel à une fonction de découverte de topologie avec multicast/broadcast, par exemple.

Quelques remarques :

*** Le paragraphe définition 3.1.1 ne sert à rien. Vous introduisez des notations qui ne serviront pas plus tard.**
Il est vrai que ces définitions ne sont plus utilisées mais elles permettent de définir les contours et les caractéristiques des réseaux auxquels nous nous attaquons.

*** "Un noeud c dispose d'un identifiantunique (ID) à partir duquel on construit les identités des autres noeuds de la communauté." S'agit-il de l'identité des autres nœuds ou de l'adresse des autres nœuds ? Dans le premier cas, la définition est récursive dans l'autre qui définit nom_communauté ?**

*** Normalement quand on ne triche pas les adresses Mac des cartes réseau sont uniques…**

Numéro du constructeur:numéro de la carte produite. Il serait plus judicieux de mettre en avant la possibilité d'héberger plusieurs noeuds et que les machines ne sont pas jetables.
Comme rappelé maintenant dans le texte, les problèmes avec les adresses MAC dupliqué sont courants. Non seulement certains fabricants "trichent" en réutilisant les adresses, mais aussi l'utilisation de machines virtuelles peut créer des adresses en double. IEEE se penche actuellement sur un nouveau modèle d'adressage qui remplacera les MAC 48 bits.

Pourquoi ne pas utilise les ID de pastry ?
L'utilisation d'un ID d'un middleware empêcherait l'utilisation de l'architecture avec d'autres middlewares. La séparation des couches (adresses GRAPP&S x adresses overlay) permet une meilleure modularité du code.

*** Je n'ai pas compris le paragraphe : "Une alternative est le mécanisme non-deterministe […] une nouvelle élection n'est nécessaire que lorsque le leader actuellement en place tombe en panne."**

Je suppose que les auteurs veulent dire que "Une alternative au mécanisme non déterministe d'OSPF est l'élection en prenant l'ID le plus fort".

Mais du coup en quoi cela limite-t-il le nombre de réélections ? (il me parait évident de ne pas refaire d'élection à tout va)

*** D'ailleurs, pourquoi décrire un processus d'élection, sans expliquer en quoi il correspond à l'architecture ?**

le texte a été retravaillé afin de clarifier ce point.

*** Remplacez panne par déconnexion volontaire ou involontaire.**

Ok

*** Vous dite que " mais reste aussi limité par leurs inconvénients, comme la dépendance aux ressources de même type ou le nombre de messages lors d'une recherche." et vous dites plus tard : "La localisation des données se fait à partir de l'identifiant du nœud DM, et par une extension des types MIME des données."**

Ai-je mal compris ou vous utilisez quelque chose que vous donnez en inconvénient ?

Ces deux affirmations ne sont pas en relation, la première indiquait une dépendance "rigide" à des types de données bien définis (fichiers, en général).

****

Cet article propose une structuration intéressante dans le cadre du pervasif pour regrouper l'information de manière hierarchique. En effet un problème primordial est la quantité d'information qu'il est impossible de rassembler en un seul lieu.

Il y a quand même quelques questions auxquelles il serait intéressant d'avoir des approfondissements:

*** Un exemple aurai put être utilisé pour montrer un cas d'utilisation hybride avec utilisation locale et globale des mêmes données**

*** Pour l'implémentation, utiliser Pastry est-il raisonnable d'un point de vue dynamicité du système ?**

*** Pour l'implémentation encore, il aurait été intéressant d'avoir un ordre de grandeur des performance et surcouts envisagés pour ce système**

L'utilisation de Pastry relève purement d'un choix pratique, vu que cela permet la création rapide d'un prototype (la gestion de connexions et d'autres tâches étant fait par Pastry). Dans un premier moment nous nous intéressons à des tests simples sur le routage et l'organisation du réseau en cas de pannes. La modularité des couches GRAPP&S doit permettre le remplacement de la couche Pastry par d'autres middlewares, afin de faire des tests de passage à l'échelle.