



MEMOIRE

Présenté à

L'École Nationale d'Ingénieurs de Sfax

en vue de l'obtention du

MASTÈRE

Dans la discipline *Informatique*
Nouvelles Technologies des Systèmes Informatiques Dédiés

Par

Fatma ABDENNADHER

(Ingénieur Informatique)

**Traitement de la Forte Dynamacité dans un Système
Publier/Souscrire Basé DHT**

Soutenu le 7 Juillet 2011, devant le jury composé de :

M. Khalil DRIRA

Président

M. Maher KHEMAKHEM

Membre

M. Wassef LOUATI

Encadreur

REMERCIEMENT

J'ai l'honneur de réserver cette page pour témoigner ma gratitude et ma reconnaissance à tous ceux qui ont contribué à ce travail.

Je tiens, tout d'abord, à remercier Monsieur Mohamed JMAIEL, Professeur à l'Ecole Nationale d'Ingénieurs de Sfax, Monsieur Wassef LOUATI, Maitre Assistant à l'Institut Supérieur d'Informatique de Mahdia, et Madame Amina CHAABANE doctorante au sein de l'unité de Recherche en Développement et Contrôle d'Applications Distribuées (ReDCAD) qui n'ont pas épargné un effort dans l'encadrement de ce projet. C'est grâce à leurs aides et leurs conseils précieux que ce travail a vu le jour.

Je tiens à exprimer mon profond respect et mes vifs remerciements envers les membres de ce jury : Monsieur Khalil DRIRA, Directeur de Recherche au Laboratoire d'Analyse et d'Architecture des Systèmes de Toulouse (LAAS-CNRS), pour l'honneur qu'il m'a fait en acceptant de le présider et Monsieur Maher KHEMA-KHEM, Maitre de conférence à l'institut supérieur de gestion à sousse, pour être rapporteur.

Enfin, j'exprime ma gratitude aux membres de ReDCAD pour l'atmosphère amicale que nous avons partagée pendant l'année dernière.

Table des matières

Introduction Générale	1
1 Les Systèmes Publier/Souscrire & les réseaux Pair-à-Pair	4
1.1 Introduction	4
1.2 Les systèmes Publier/Souscrire	5
1.2.1 Architecture	7
1.2.2 Filtrage	7
1.2.3 Type de messages	8
1.2.4 Caractéristiques	8
1.2.5 Classification des systèmes Publier/Souscrire	10
1.2.5.1 Classification des systèmes selon le langage de sous- scription	10
1.2.5.2 Classification selon la topologie de l'infrastructure Publier/Souscrire	11
1.3 Les Réseaux Pair-à-Pair	15
1.3.1 Définition	15
1.3.2 Domaines d'application des réseaux Pair-à-Pair	16
1.3.3 Caractéristiques des réseaux Pair-à-Pair	16
1.3.4 Classification des réseaux Pair-à-Pair	18
1.3.4.1 Les réseaux Pair-à-Pair non structurés	18
1.3.4.2 Les réseaux Pair-à-Pair structurés : Les tables de hachage distribuées	18
1.4 Conclusion	23

2	Systèmes publier/souscrire basés DHT & dynamisme	25
2.1	Introduction	25
2.2	Dynamicit� des arbres	25
2.3	Exemples DHT et Publier/Souscrire	26
2.3.1	La DHT Pastry	26
2.3.2	Le syst�me Publier/Souscrire Scribe	30
2.4	Notion du churn	31
2.5	Probl�me du churn	31
2.6	Effet du churn sur les performances des DHTs	32
2.7	Traitement du churn dans la DHT	32
2.7.1	La R�plication dans les DHTs	32
2.7.1.1	Protocole de placement Initial	32
2.7.1.2	Protocole de maintenance	34
2.7.2	Relaxation des contraintes de placement de la DHT pour tol�rer le churn	35
2.8	Conclusion	35
3	Conception et �valuation de l'approche propos�e	37
3.1	Introduction	37
3.2	Approche propos�e	37
3.2.1	Justification du choix du syst�me	38
3.2.1.1	Pastry	38
3.2.1.2	Scribe	38
3.2.1.3	P2P-TOPSS	38
3.2.2	Concept P-Grid	39
3.2.3	Sp�cification de notre approche de r�f�rencement	40
3.2.3.1	Phase de souscription	40
3.2.3.2	Phase de publication	43
3.3	Impl�mentation et Evaluation	46
3.3.1	FreePastry	46
3.3.2	Les m�triques mesur�es	47

3.3.2.1	Taux de churn	47
3.3.2.2	Taux de réussite de matching	48
3.3.2.3	Trafic de messages	48
3.3.2.4	Taux d'erreur de matching	48
3.3.3	Scénario	49
3.3.4	Résultats et interprétations	50
3.4	Conclusion	57
	Conclusion Générale	58
	Bibliographie	60

Table des figures

1.1	Modèle de communication du système Publier/Souscrire	5
1.2	Modèle architectural du système Publier/Souscrire	7
1.3	Découplage dans le temps entre producteurs et consommateurs	8
1.4	Découplage dans l'espace entre producteurs et consommateurs	9
1.5	Découplage de synchronisation entre producteurs et consommateurs	9
1.6	Topologie hiérarchique	11
1.7	Topologie Pair-à-Pair acyclique	12
1.8	Topologie Pair-à-Pair générique	13
1.9	Topologie hybride : hiérarchique, générique	14
1.10	Topologie hybride : acyclique, générique	15
2.1	Routage vers la clé (d46a1c)	27
2.2	Table de routage du pair 10233102	28
2.3	Arbre de Scribe	30
2.4	Réplication basée sur le voisinage	33
2.5	Réplication à base de clés multiples	34
3.1	Arbre de P-Grid	39
3.2	Exemple de référencement	41
3.3	Exemple 1 de publication	44
3.4	Exemple 2 de publication	45
3.5	Exemple 3 de publication	46
3.6	Dégradation de la disponibilité des souscriptions en fonction de l'augmentation du taux de churn	50
3.7	Référencement aléatoire	51

3.8	Comparaison du trafic moyen de messages en fonction du taux de churn	52
3.9	Comparaison du taux moyen de réussite en fonction du taux de churn	53
3.10	Taux moyen de réussite en fonction du nombre de références	54
3.11	Trafic moyen de messages en fonction du taux de réussite	55
3.12	Taux moyen d'erreur en fonction du nombre de références	56
3.13	Comparaison du taux d'erreur observé en fonction du taux de churn .	56

Liste des tableaux

1.1 Performances des différentes tables de hachage distribuées	22
--	----

INTRODUCTION GÉNÉRALE

Depuis les débuts du réseau Internet, le modèle client-serveur présentait le modèle de base pour mettre à disposition des ressources. Dans ce paradigme, le système dispose d'un serveur dédié qui centralise et supporte l'ensemble des services et des ressources. Dès ce moment-là, l'accroissement du nombre d'utilisateurs exige que les fournisseurs de services investissent de plus. Il est en effet indispensable d'assurer la disponibilité des ressources et des services, même face à un grand nombre de demandes simultanées. Ceci requiert d'importantes ressources et impose des contraintes logicielles et matérielles, ce qui rend de tels systèmes très coûteux. Pour palier à ce problème le paradigme Publier/Souscrire a été donc proposé comme abstraction possible pour la programmation répartie afin de répondre aux besoins d'extensibilité des applications courantes.

Ainsi, depuis le choix du modèle Publier/Souscrire comme étant une méthode puissante de diffusion d'informations, plusieurs études de recherche furent lancées dans ce sujet pour concevoir des systèmes Publier/Souscrire plus performants et adaptés aux différentes applications réelles. Particulièrement leur déploiement sur les services d'événements Pair-à-Pair de forte mobilité. En effet, ce type de réseau est caractérisé par une topologie dynamique, ce qui nécessite la mise en œuvre de stratégie permettant l'optimisation des performances de ces systèmes.

Toutefois, le modèle Pair-à-Pair est décentralisé et ne dispose pas d'une entité centrale pour coordonner l'interconnexion des pairs et s'arranger selon une topologie dynamique permettant d'assurer un routage efficace. C'est pour cela que des systèmes qui imposent une structure entre les pairs sont évoqués afin de garantir un diamètre optimal. Parmi ces systèmes, nous citons les structures basées sur le principe des tables de hachage distribuées (DHT). Le principe est de structurer les pairs selon un réseau logique structuré par exemple un anneau ou un hyper-cube, dans le but d'employer des techniques de routage efficaces. Ces structures sont caractérisées par une administration transparente, mais sont complexes dans leur conception.

De nombreux défis complexes ont été soulevés par la recherche sur les réseaux Pair-à-Pair tels que l'équilibre de charge dans le réseau, le passage à l'échelle, la minimisation des coûts, le routage, la sécurité, la gestion de la dynamique des nœuds au niveau du réseau (le va des nœuds et le vient de nouveaux nœuds "churn"), etc. Particulièrement, le problème de la gestion de la forte dynamique des nœuds, a soulevé un vif intérêt dans la communauté des chercheurs ce qui a induit à l'apparition de plusieurs solutions. Bien que ces implémentations montrent certaines caractéristiques pertinentes, ils dévoilent certaines limites, surtout à forte dynamique des entités du réseau. En fait, nous sommes face à une connexion de nouveaux pairs du réseau Pair-à-Pair et une déconnexion des pairs connectés à tout instant. De ce fait, telles ruptures brusques dans le réseau causent la perte des données et la perturbation des tables de routage. Il est donc important de concevoir des stratégies de résistance au churn pour faire face à ces changements brusques dans la structure du réseau.

C'est dans cette perspective que se situe notre projet de mastère, réalisé au sein de l'unité de Recherche en Développement et Contrôle d'Applications Distribuées (ReDCAD), qui consiste à proposer une nouvelle approche de résistance au churn, appliquée sur un système Publier/Souscrire basé sujet et structuré au dessus d'une table de hachage distribuée. Notre approche de résistance au churn est basée sur le référencement à base de digit. Notre référencement permet de garantir la disponibilité des souscriptions au sein du service d'événements même à un dynamisme fort tout en minimisant le trafic de messages.

Ce rapport, développant le thème de la résistance au churn dans les systèmes Publier/Souscrire basés DHT, est organisé suivant trois chapitres : Le premier chapitre est consacré, dans une première partie, à la présentation des systèmes Publier/Souscrire. Dans une deuxième partie, il présente le principe du paradigme Pair-à-Pair en présentant une classification des grandes catégories des systèmes existants. Nous détaillons en particulier la catégorie des systèmes Pair-à-Pair structurés, basés sur les DHTs qui présente la structure de base pour notre système Publier/Souscrire. Dans le deuxième chapitre, nous combinons la notion du churn avec les systèmes publier/-souscrire basés DHT en passant en revue quelques exemples de ces systèmes. Par

la suite, nous présentons l'effet du churn sur les performances des DHTs. Ensuite, nous exposons un aperçu sur l'état de l'art des différentes solutions basées sur la réplication afin d'améliorer la disponibilité des données au sein du réseau. Nous introduisons, dans le troisième chapitre, notre nouvelle approche de résistance au churn dans les systèmes Publier/Souscrire basés sur les DHTs, utilisant le référencement. Cette approche est ultérieurement implémentée au niveau du système P2P-TOPSS [TAJ03] qui est un système Publier/Souscrire basé contenu développé au dessus de Scribe [CDKR02] en intégrant sur ce dernier certaines modifications. Par la suite, nous passons à l'évaluation des performances de notre approche par la réalisation d'un ensemble de tests. Nous clôturons finalement notre rapport par une conclusion et des perspectives.

Chapitre 1

Les Systèmes Publier/Souscrire & les réseaux Pair-à-Pair

1.1 Introduction

Actuellement la communication consomme beaucoup de ressources réseau. Le paradigme de communication dans ces applications est de type client/serveur, où la communication a lieu typiquement entre deux entités, le client et le serveur. Les clients de l'application partagent une unique instance du monde virtuel au moyen du serveur, qui leur passe, après les avoir traité, tous les événements issus de tous les clients. Or, un participant n'a besoin de recevoir que les événements qui lui sont pertinents. Par conséquent, ils doivent eux-mêmes être responsables de filtrer les événements pertinents qui leur arrivent.

Comme solution aux problèmes posés par le paradigme client/serveur, les réseaux Pair-à-Pair sont apparus pour permettre une propagation de l'information dans un système distribué appartenant à un réseau à large échelle. Dans ce cadre, plusieurs travaux de recherche tentent à assurer l'utilisation des systèmes publier/souscrire sur les réseaux Pair-à-Pair.

La première partie de ce chapitre est destinée à la présentation des systèmes Publier/Souscrire, leur architecture, leurs principales caractéristiques et leurs différentes classifications. Dans la deuxième partie, nous présentons les réseaux Pair-à-Pair en détaillant leurs domaines d'applications, leurs caractéristiques, leurs dis-

tinctes classifications .Nous clôturons le chapitre par la focalisation sur les tables de hachage distribuées de classe réseaux Pair-à-Pair structurés puisque cette gamme de réseaux Pair-à-Pair présente le champs de notre recherche.

1.2 Les systèmes Publier/Souscrire

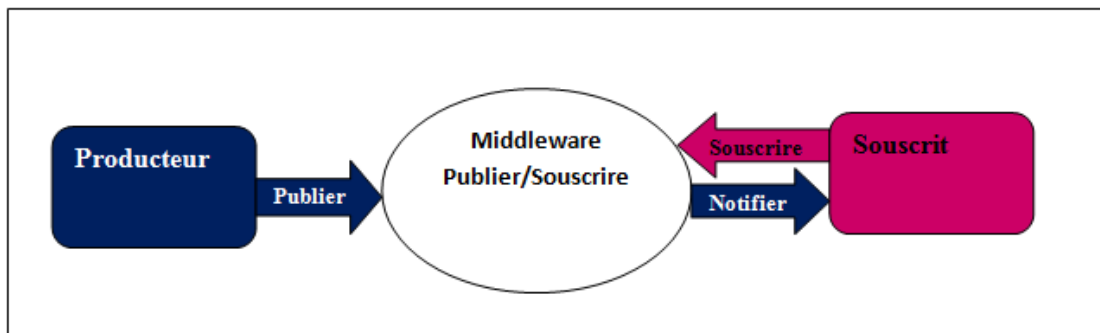


FIG. 1.1 – Modèle de communication du système Publier/Souscrire

Ces systèmes sont de nouveaux systèmes de communication qui permettent aux fournisseurs de l'information (producteur) d'envoyer des notifications aux récepteurs de l'information (souscrit) à travers les serveurs d'évènements comme le montre la figure 1.1.

Les applications distribuées exploitant les systèmes Publier/Souscrire sont organisées comme une collection de composants autonomes (clients) qui interagissent en publiant des messages et en souscrivant à des catégories de messages qui les intéressent. Le composant de base du service d'évènements est le dispatcher qui est responsable du filtrage et routage des évènements. Il est chargé de recueillir des souscriptions sous forme de filtre à partir des souscrits. Lors de la réception de la publication, il notifie uniquement les utilisateurs intéressés selon les souscriptions déjà mémorisées. Il en résulte un découplage fort entre les parties communicantes.

- découplage dans l'espace : les parties communicantes n'ont pas besoin d'être connectées ou même connaître les unes les autres
- découplage dans les flux : les parties communicantes n'ont pas besoin d'être synchronisées

- découplage dans le temps : les parties communicantes n'ont pas besoin d'être connectées en même temps

Les systèmes Publier/Souscrire sont différenciés par l'expressivité du langage des souscriptions. Nous distinguons les systèmes basés type, sujet et contenu. Dans le premier cas, le filtrage spécifie le type de l'événement. Dans le deuxième cas, les souscriptions ne contiennent que le nom d'une classe de messages - généralement appelé le canal ou le sujet - choisi parmi un ensemble de classes prédéfinies. Dans le troisième cas, la sélection d'un message est entièrement déterminée par le client, qui utilise des expressions (souvent appelés filtres) qui permettent une correspondance sophistiquée sur le contenu des messages.

Le deuxième point de différenciation est l'architecture du service d'événements, qui peut être centralisée ou distribuée. Dans ce dernier type, un ensemble de dispatchers est interconnecté à un réseau de recouvrement ; ils routent en collaboration les souscriptions et les messages envoyés par les clients qui y sont connectés, ce qui favorise l'évolutivité du système. Ainsi, l'architecture des systèmes Publier/Souscrire est bien spécifique. En effet, les fonctionnalités offertes par ces systèmes requièrent des opérations de filtrage et de routage bien spécifiques.

1.2.1 Architecture

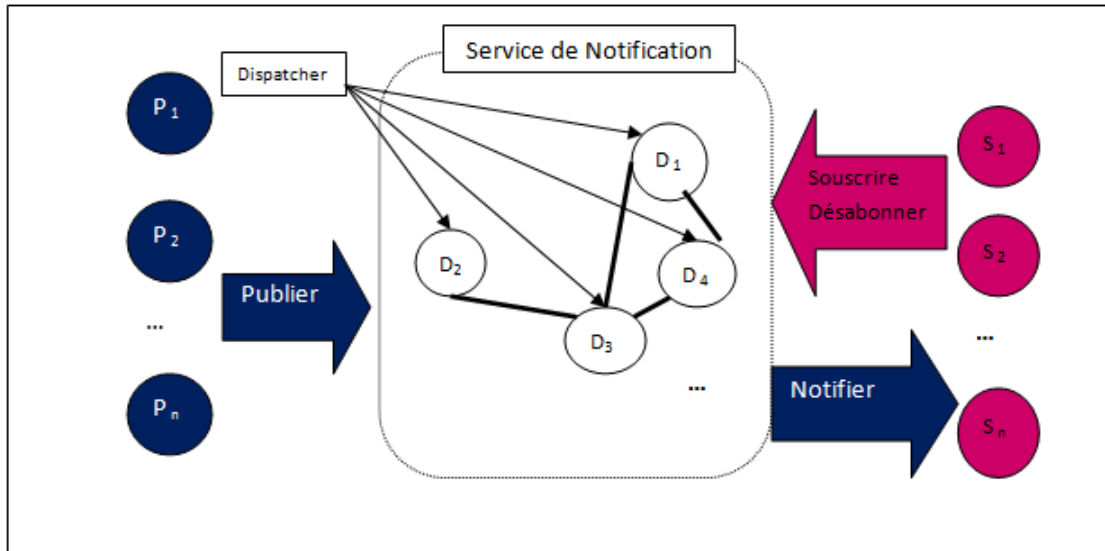


FIG. 1.2 – Modèle architectural du système Publier/Souscrire

D'après la figure 1.2 les procédés dans les systèmes Publier/Souscrire sont des clients d'un service de notification sous-jacent et peuvent agir en tant que producteurs et consommateurs de messages. L'interface de communication vers le système se compose de quatre primitives seulement : publier, souscrire, désabonner, et notifier.

1.2.2 Filtrage

Un filtre est un outil d'expression du besoin du consommateur. Il est formé d'un ensemble d'attributs qui spécifient les contraintes des exigences du consommateur et chaque contrainte présente un quadruplet (type, nom, valeur, opérateur). Le régime le plus souple est proposé par le filtrage basé contenu. Les filtres sont des fonctions booléennes sur la totalité du contenu d'une notification et une voie commune pour mettre en œuvre les souscriptions.

1.2.3 Type de messages

Les participants à une communication de type Publier/Souscrire se comportent comme des consommateurs qui expriment leurs besoins et leurs demandes par souscription dans le service d'événements afin de récupérer les notifications qui répondent à leurs besoins. De l'autre partie du réseau, les producteurs produisent des événements et les publient au service d'événements. Ce dernier prend en charge l'association (matching) des publications aux souscriptions des clients et finit par acheminer les notifications uniquement aux clients intéressés. Une notification est un message résultant du filtrage souscription/publication pour notifier les souscrits afin de les informer qu'un événement est prêt d'être récupéré. Les notifications sont acheminées par le service de notification à ces consommateurs qui ont inscrit une souscription correspondante. Les souscriptions décrivent le type de notifications que les consommateurs sont intéressés. Dans certains systèmes les producteurs sont tenus de délivrer des publicités pour décrire les publications prévues d'être publiées.

1.2.4 Caractéristiques

Le découplage tridimensionnel est une caractéristique du modèle Publier/Souscrire :

- Découplage dans le temps

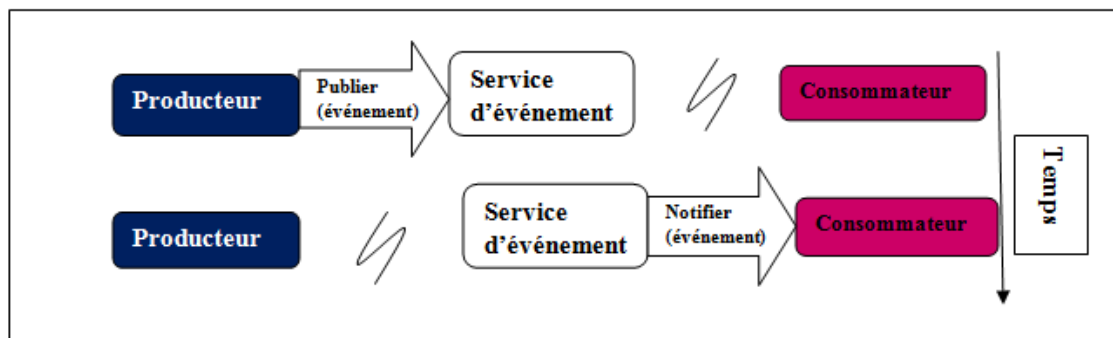


FIG. 1.3 – Découplage dans le temps entre producteurs et consommateurs

La Figure 1.3 illustre que la connexion en même temps des participants associés n'est pas exigée. En particulier, la production des événements par les

producteurs se déroule normale même pendant la déconnexion des consommateurs. De même, la notification des consommateurs se fait même pendant la déconnexion des producteurs.

- **Découplage dans l'espace**

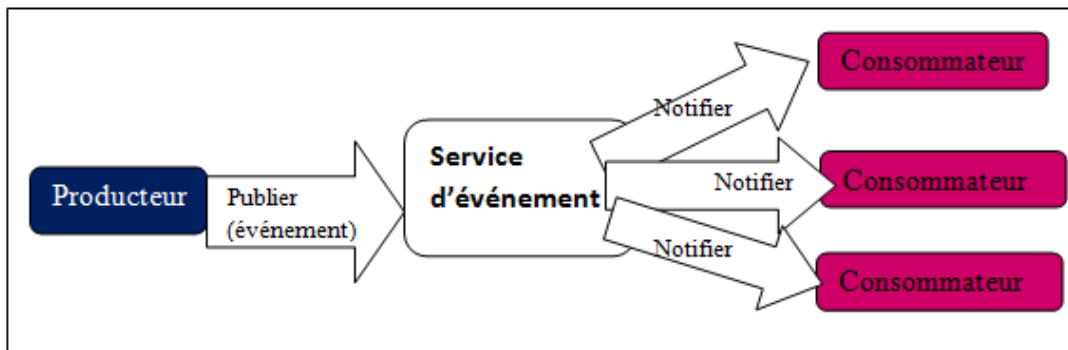


FIG. 1.4 – Découplage dans l'espace entre producteurs et consommateurs

D'après la figure 1.4, le producteur et le consommateur entrent en interaction sans avoir besoin que l'un connaisse l'autre. Ainsi, la production des événements par les producteurs se déroule dans le service d'évènements et les consommateurs y accèdent pour les consommer sans aucune connaissance concernant la source de ces évènements, et le nombre de producteurs.

- **Découplage de synchronisation**

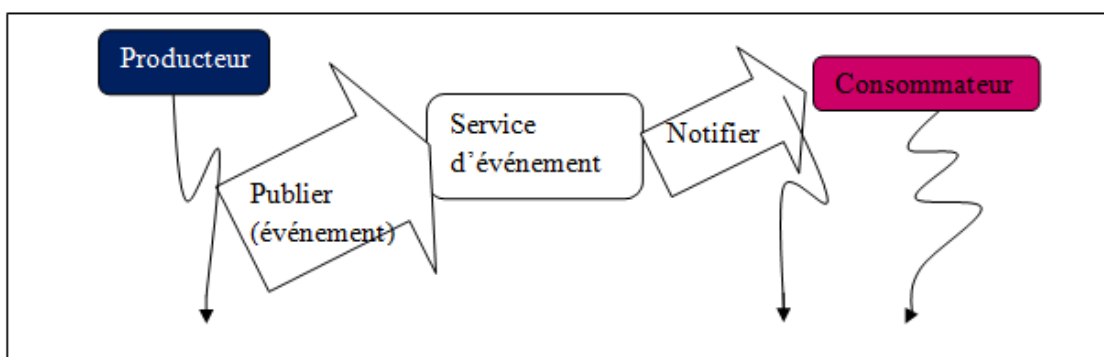


FIG. 1.5 – Découplage de synchronisation entre producteurs et consommateurs

Dans le découplage illustré par la figure 1.5, la notification des consommateurs se déroule en même temps que la production d'autres notifications par les producteurs sans synchroniser entre ces évènements. Ceci montre l'indépendance

totale des parties participantes à cette communication.

Ainsi, ce découplage tridimensionnel qui caractérise le modèle Publier/Souscrire assure la facilité de la communication entre les producteurs et les consommateurs et favorise la scalabilité de ces systèmes.

- **Scalabilité** : Un middleware scalable a la capacité de supporter un nombre énorme de clients et de serveurs. Ce qui représente une caractéristique importante pour les applications distribuées sur Internet vu le grand nombre de producteurs et de consommateurs participants.
- **Expressivité** : Les systèmes Publier/Souscrire se caractérisent par un modèle expressif de spécifications d'évènements. De ce fait, les souscriptions des systèmes basés contenu expriment bien les besoins des consommateurs.

1.2.5 Classification des systèmes Publier/Souscrire

La classification des systèmes Publier/Souscrire peut être selon le langage de la souscription ou bien selon la topologie des dispatchers au sein du service d'évènements.

1.2.5.1 Classification des systèmes selon le langage de souscription

Chaque consommateur exprime ses intérêts en spécifiant les événements qu'il voulait bien recevoir, la méthode de spécification des événements diffère d'un système à un autre selon le langage de souscription adopté. Ainsi nous obtenons la classification ci-dessous des systèmes Publier/Souscrire.

- **Système basé sujet** : Le système basé sujet représente le plus ancien système basé événement et il est implémenté par divers systèmes. Comme exemple de systèmes basés sujet nous citons SCRIBE [CDKR02], TIB/RV [OPSS93] et Bayeux [ZZJ⁺01].
- **Système basé type** : Dans le cas où le type de l'évènement envoyé est simple alors le filtrage spécifie s'il s'agit d'une chaîne de caractères, d'un entier, d'un entier long ou d'un double. Dans le cas où l'évènement envoyé est un objet le filtrage revient à trouver la classe de l'objet.

- **Système basé contenu** : Le système basé contenu, permet aux clients une souscription sur le contenu de l'information bientôt publiée par un producteur. Ce mode de communication a bien résolu la limitation d'expressivité pour les deux premiers types de systèmes. Mais l'implémentation pour ce modèle est plus compliquée. Afin que la diffusion des messages soit correcte, une fonction de correspondance (matching) est construite pour associer correctement les messages en provenance des producteurs aux souscrits. Parmi les systèmes basés contenu, nous trouvons SIENA [CRW01] et JEDI [CNF98].

1.2.5.2 Classification selon la topologie de l'infrastructure Publier/Souscrire

Cette classification se base sur la topologie de la connexion entre les dispatchers. Ainsi quatre types de topologies sont distingués : client/serveur hiérarchique, Pair-à-Pair acyclique, Pair-à-Pair générique, et topologie hybride.

- **Topologie client/serveur hiérarchique**

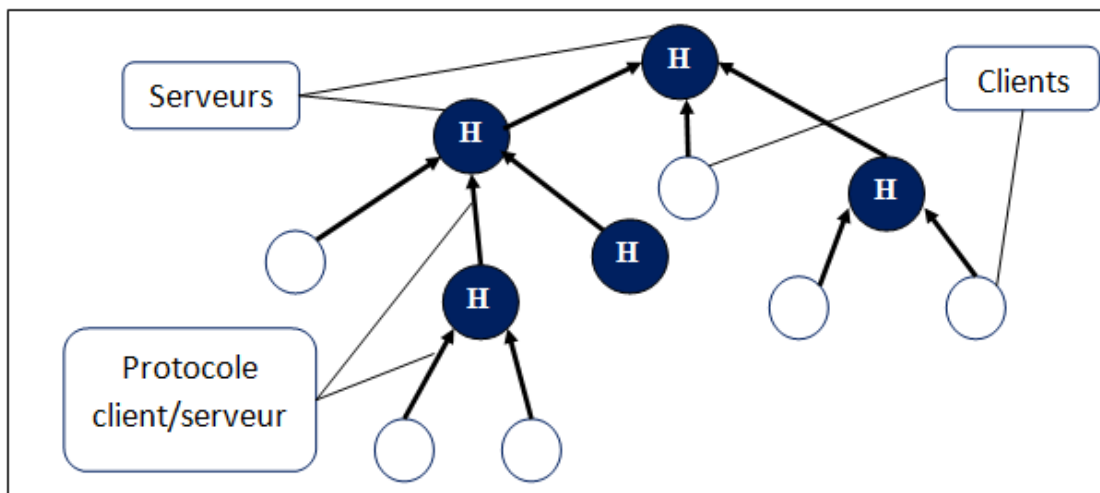


FIG. 1.6 – Topologie hiérarchique

Dans cette topologie représentée sur la figure 1.6, les pairs reliés s'interagissent au sein d'une relation asymétrique de client/serveur. Chaque serveur se relie à un serveur père et fournisse des services pour certains clients. Le serveur racine n'est pas relié à un serveur père. Cette topologie représente bien une extension

de l'architecture client-serveur. Le point de défaillance dans une telle topologie est que la panne d'un serveur engendre le détachement de tous les serveurs de ses sous arbres.

- **Topologie Pair-à-Pair acyclique**

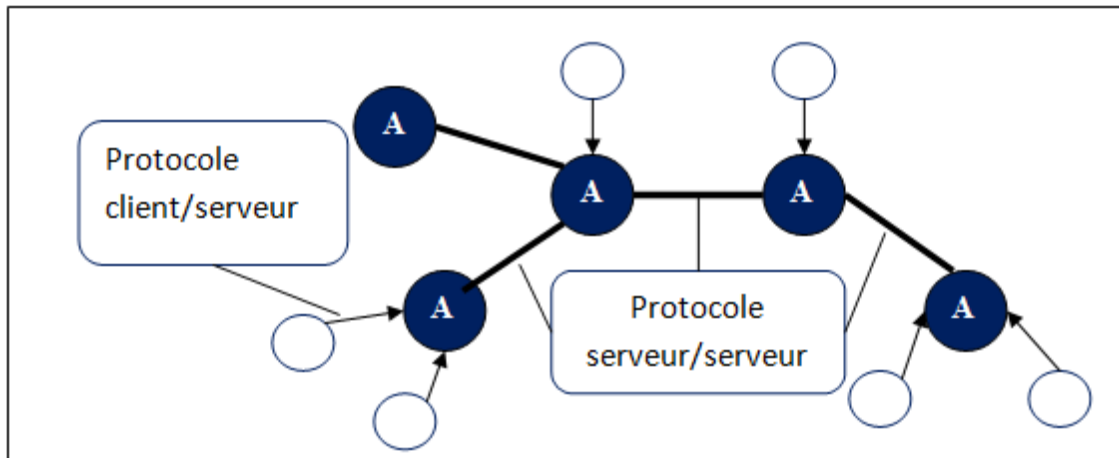


FIG. 1.7 – Topologie Pair-à-Pair acyclique

La topologie Pair-à-Pair acyclique représente une extension de la topologie hiérarchique. Un graphe acyclique est formé par les connexions entre les serveurs, donc la redondance de chemins est éliminée ce qui induit à l'unicité du chemin entre les serveurs. La communication entre les serveurs adopte un protocole bien précis permettant un flux bidirectionnel de souscriptions, et de notifications. Les lignes non orientées représentent la communication entre les serveurs alors que les lignes orientées représentent la communication clients/-serveurs comme montré dans la figure 1.7. La mise en œuvre de la topologie acyclique dans un système distribué à large échelle est difficile. De plus, l'échec d'un serveur conduit à la déconnexion de tous les serveurs qui lui sont connectés.

- **Topologie Pair-à-Pair générique**

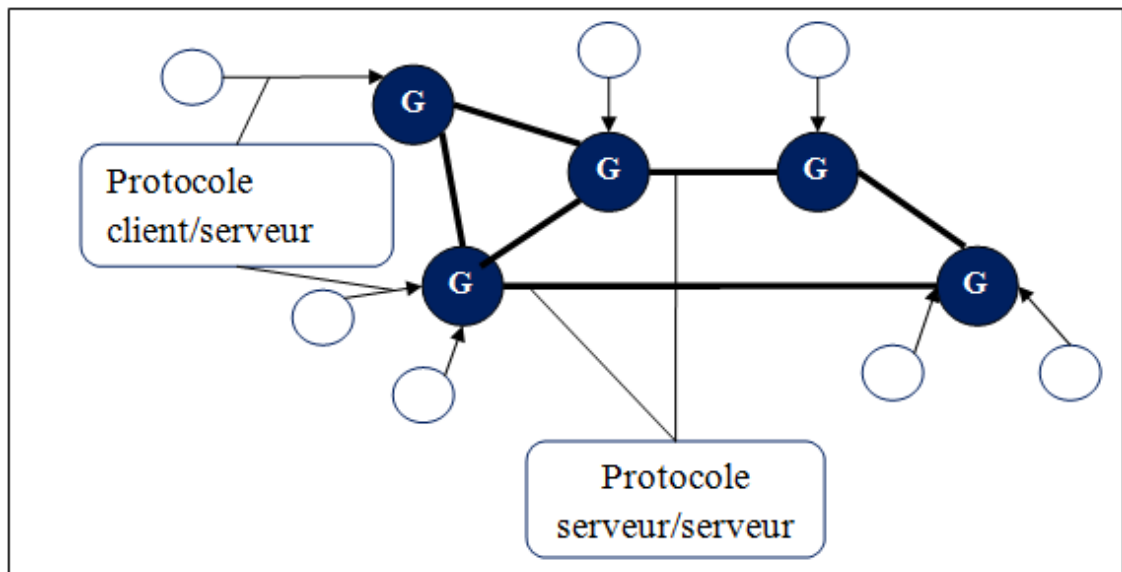


FIG. 1.8 – Topologie Pair-à-Pair générique

Cette topologie montrée par la figure 1.8 assure une communication symétrique, bidirectionnelle entre deux serveurs en échange des souscriptions et des notifications. Ainsi, il existe différents chemins qui relient deux serveurs de cette topologie. L'apport de cette topologie, en comparaison avec la topologie Pair-à-Pair acyclique, est la flexibilité, et la robustesse grâce à la redondance de chemin entre les nœuds du réseau. Mais en contre partie, elle nécessite un algorithme de routage très précis.

- **Topologie hybride**

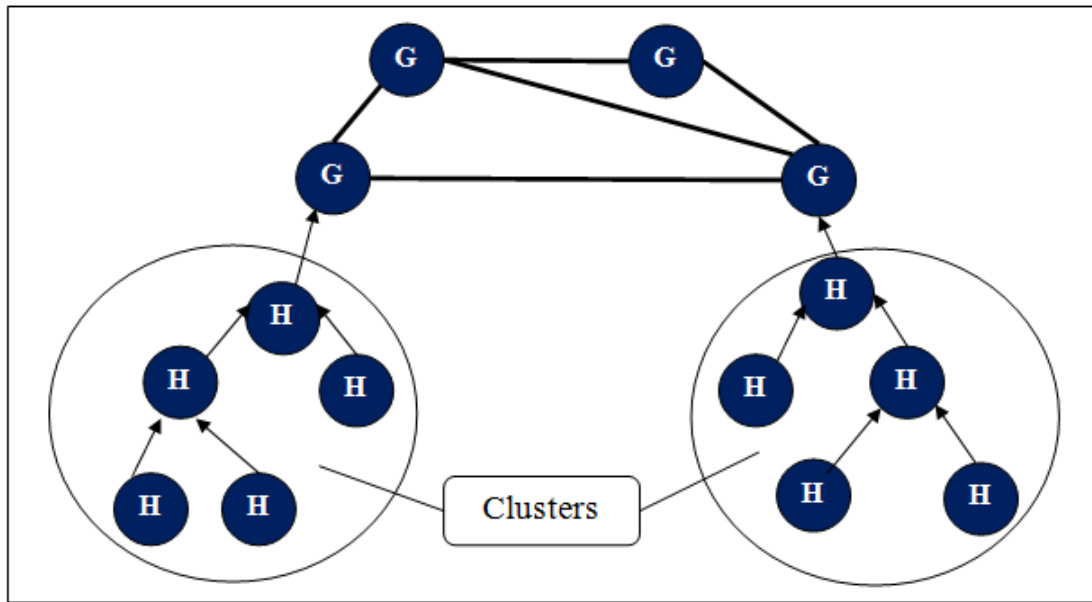


FIG. 1.9 – Topologie hybride : hiérarchique, générique

La topologie hybride représentée par la Figure 1.9 regroupe les deux topologies hiérarchique et générique ce qui assure une robustesse et une scalabilité considérables. Comme illustré par la Figure 1.9, un cluster est constitué d'un ensemble de serveurs liés selon une topologie hiérarchique. L'ensemble de ces clusters forme l'hiérarchie Pair-à-Pair générique. Un niveau de contrôle et de coordination excellent doit exister au sein d'un même cluster.

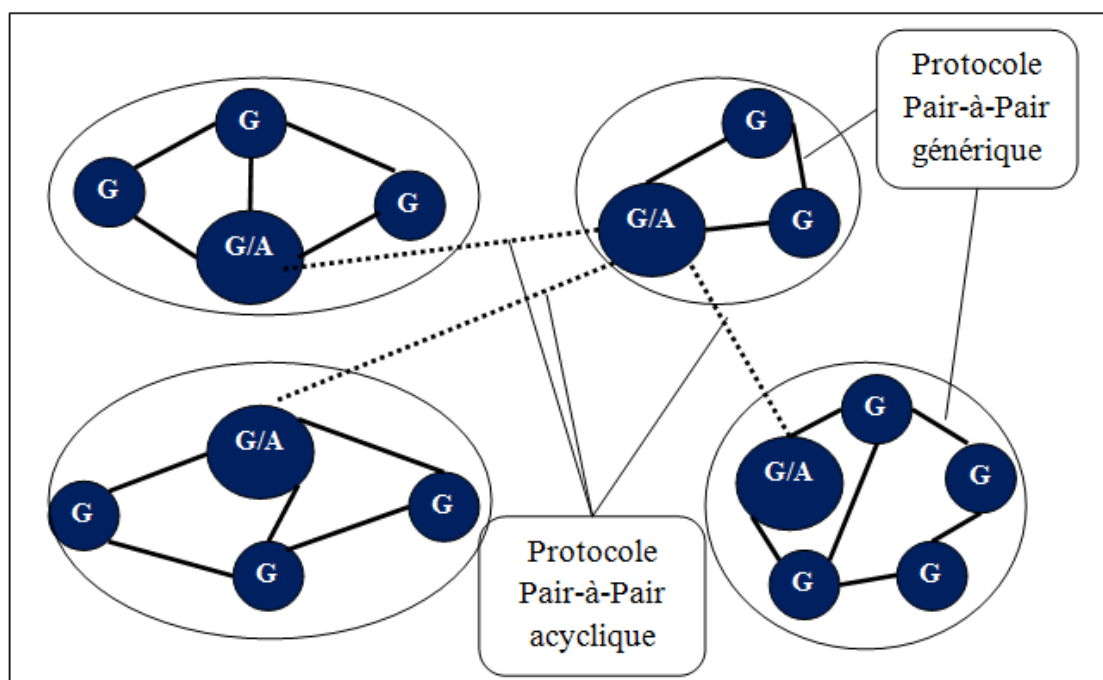


FIG. 1.10 – Topologie hybride : acyclique, générique

Comme illustré par la Figure 1.10, dans la Topologie hybride (acyclique, générique) un cluster est constitué d'un ensemble de serveurs liés selon une topologie générique. L'ensemble de ces clusters forme la topologie Pair-à-Pair acyclique.

1.3 Les Réseaux Pair-à-Pair

1.3.1 Définition

Un réseau Pair-à-Pair est un réseau distribué constitué de nœuds présentant des rôles et des usages égaux qui échangent entre eux des informations et des services [YGM01]. Cette définition présente le fondement de la réalité du contexte d'utilisation de ces réseaux. Ainsi, il n'y a aucune entité centrale qui maîtrise le réseau. Ce qui entraîne un certain nombre d'avantages, mais aussi des exigences. Pour les avantages, l'augmentation du nombre des pairs augmente les ressources partagées tout en distribuant la charge entre les pairs. Ce qui permet un passage à l'échelle avec un coût relativement faible. De plus, la caractéristique de distribution contri-

bue à la robustesse du système, du fait que la disponibilité des ressources ne dépend pas seulement d'un pair en particulier [DGMP08]. Pour les exigences, la mise en place d'un mécanisme d'auto-organisation devient obligatoire puisqu'il n'y a plus un élément central. La connexion et la déconnexion des pairs au réseau doivent être effectuées sans affecter la disponibilité des ressources.

1.3.2 Domaines d'application des réseaux Pair-à-Pair

Il existe de nombreux domaines applicatifs liés aux réseaux Pair-à-Pair. Parmi les principaux nous trouvons les applications distribuées de partage de fichiers comme Gnutella [AH00], Kazaa¹ et Freenet², qui ont largement participé au développement du Pair-à-Pair. Nous citons aussi les applications de communication Pair-à-Pair comme la messagerie instantanée tel que ICQ³ et la voix sur IP comme Skype⁴. De plus, les applications de calcul intensif distribué comme Seti@home⁵ ont utilisé la répartition du Pair-à-Pair pour répartir ce calcul sur de nombreux processeurs disponibles à travers le réseau. La distribution des données et des traitements sur tous les pairs du réseau, assure un grand passage à l'échelle sans l'utilisation de serveurs surpuissants. De même, le développement des jeux à multi joueurs à large échelle profite bien des caractéristiques des réseaux Pair-à-Pair. Aussi, les applications collaboratives telles que Pastis [BPS05], Oceanstore [KBC⁺00], Ivy [MMGC02], gagnent de la distribution du Pair-à-Pair dans le but d'accorder la coopération d'un très grand nombre de collaborateurs qui travaillent sans coordination pour l'élaboration d'une ressource.

1.3.3 Caractéristiques des réseaux Pair-à-Pair

D'après [HAY⁺05] les caractéristiques des réseaux Pair-à-Pair sont :

- **Scalabilité** : la montée en échelle du nombre de pairs atteignant l'ordre de millions est bien supportée par les systèmes Pair-à-Pair.

¹<http://www.kazaa.com/>

²<http://freenetproject.org/>

³<http://www.icq.com/>

⁴<http://www.Skype.com/>

⁵<http://setiathome.ssl.berkeley.edu/>

- **Volatilité des nœuds** : les pairs se connectent et se déconnectent imprévisiblement de façon que la structure du réseau soit instable. Le système devrait résister à ce phénomène appelé "churn" et faire les mises à jour appropriées afin de s'auto-organiser et garantir la disponibilité des ressources dans un système distribué pouvant être face à des pannes et des départs de nœuds à tout moment.
- **Localisation rapide des ressources dans un environnement distribué** : Le mécanisme de localisation des ressources distribuées sur les différents pairs devrait être efficace et capable de s'auto-adapter aux changements perpétuels de la topologie du réseau.
- **Autonomie des pairs** : L'absence de l'entité centrale de contrôle oblige chaque participant au réseau Pair-à-Pair qu'il ait une vue partielle du réseau, aucun pair n'est capable de connaître ou contrôler tout le système. Le système devrait être assez robuste pour tolérer les échecs et les déconnexions fréquentes des nœuds du réseau.
- **Hétérogénéité du réseau selon les pairs** : les pairs du réseau peuvent être de nature hétérogène (architecture matérielle : processeur, mémoire et type de connexion : modem, haut débit) exemple : PC, PDA, téléphone, etc.
- **Décentralisation** : les systèmes Pair-à-Pair sont décentralisés ce qui leur permet d'avoir des mécanismes supportant le stockage distribué, le partage d'informations, le traitement distribué, etc.
- **Symétrie** : la symétrie apparaît dans les rôles des pairs participants. Les systèmes classiques client/serveur sont asymétriques et les serveurs sont souvent plus puissants que les clients. Par contre, dans les systèmes Pair-à-Pair, les nœuds sont symétriques en jouant le rôle de serveur et de client à la fois, de ce fait un pair peut être producteur d'une ressource et consommateur d'une autre ressource dans le réseau.
- **Equilibre de charge entre les nœuds du réseau** : le système doit optimiser la distribution optimale des ressources, basée sur la capacité et la disponibilité des ressources de chaque pair.

1.3.4 Classification des réseaux Pair-à-Pair

1.3.4.1 Les réseaux Pair-à-Pair non structurés

- **Les architectures hybrides** : Les architectures hybrides présentent la première génération des réseaux Pair-à-Pair. Pour ce type d'architecture, les pairs sont hiérarchisés par le réseau selon leurs capacités. Les informations de localisation des données sont maintenues dans un serveur annuaire central. En effet, chaque nœud, qui joint le système, se connecte au serveur et annonce le sujet de ses fichiers partagés. Par la suite, les serveurs sont contactés pour obtenir les informations de localisation potentielle (identité du pair sur lequel sont stockées les données à chercher).
- **Les architectures basées sur la diffusion** : Les architectures basées sur la diffusion sont considérées comme la deuxième génération des réseaux Pair-à-Pair. Dans ce type d'architecture il n'y a pas de règles de connexion imposées entre les pairs participants. Il n'y a aucune information sur la localisation des données dans le système. La recherche se déroule par inondation dans le réseau par les voisins selon ce processus : le nœud source de la requête diffuse le message de recherche à tous ses pairs voisins. Ces pairs voisins, rediffusent à leurs tours le message reçu et le processus se poursuit jusqu'à trouver la donnée désirée ou bien si le TTL (Time To Live) de la requête soit atteint, comme exemple de ce type d'architecture nous citons le protocole Gnutella [AH00]. Ces deux architectures non structurés présentent des limites. Dans Les systèmes hybrides, le serveur risque d'être exposé à la défaillance, à l'attaque ainsi qu'au goulot d'étranglement. Alors que pour les systèmes basés sur la diffusion, une consommation élevée de la bande passante est constatée. Les tables de hachage distribuées (DHT), appelées aussi les réseaux Pair-à-Pair structurés éliminent ces limites et permettent le passage à l'échelle.

1.3.4.2 Les réseaux Pair-à-Pair structurés : Les tables de hachage distribuées

- **principe général**

Les réseaux Pair-à-Pair structurés ont vu le jour principalement dans le but de répondre aux problèmes d'évolutivité que les systèmes non structurés ne parviennent pas à résoudre. Contrairement aux systèmes Pair-à-Pair non structurés, dans les systèmes Pair-à-Pair structurés, les pairs sont logiquement placés dans des locations déterministes pour former et maintenir une structure spéciale grâce à une topologie du réseau contrôlé.

Les tables de hachage distribuées forment la classe la plus importante des systèmes Pair-à-Pair structurés. Dans une DHT, chaque nœud est titulaire d'une partie de la table de hachage et un objet est stocké sur ce nœud si l'identifiant de l'objet appartient à la gamme dont le nœud est responsable.

Les DHTs ont un espace d'identifiants formé par les identifiants des nœuds (nodeIds) et les identifiants des objets (objectIds (clés)) et ces identifiants sont uniformément générés. Étant donné une clé, put(clé, objet) stocke l'objet vers le nœud correspondant à la clé. Pour récupérer un objet avec un identifiant donné clé, get(clé) est appelée. Chacune de ces deux opérations ont besoin d'un algorithme de routage pour envoyer la demande au pair chargé de la clé.

Récemment, les tables de hachage distribuées (DHT) ont émergé comme une infrastructure pour l'efficacité de la recherche des ressources évolutives dans des réseaux Pair-à-Pair distribués. De tels systèmes sont décentralisés, évolutives, et auto organisables (ils s'adaptent automatiquement à l'arrivée, le départ des nœuds dans le réseau). De telles caractéristiques rendent les DHTs intéressantes pour créer des applications distribuées. En fait, les tables de hachage distribuées ont été utilisées avec succès dans plusieurs domaines d'application, tels que les systèmes de sauvegarde, de gestion distribuée de fichiers, ou les systèmes de distribution de données. Il a également été montré que les systèmes Publier/Souscrire basés sujet peuvent être construits au-dessus d'une DHT.

Les systèmes Pair-à-Pair basés sur des DHTs sont structurés suivant trois couches :

- 1) La couche de routage : La couche de routage appelée aussi KBR (Key-Based Routing) puisqu'elle utilise des clés pour identifier les pairs. Cette couche per-

met de masquer la complexité de l'infrastructure, des algorithmes de tolérance aux fautes et du routage et pour les couches supérieures. Actuellement, des améliorations de la résistance de cette couche aux forts taux de churn ont été menées. Les principaux exemples de la couche de routage sont Pastry [RD01], Chord [SMLN⁺03], Tapestry [ZHS⁺04] et CAN [RFH⁺01].

2) La DHT en elle-même : La couche DHT stocke des blocs de données grâce à un service de stockage distribué, persistant, tolérant aux fautes, et scalable. Les DHTs fournissent les primitives get et put, pour simplifier la conception d'applications à large échelle. PAST et DHash sont des DHTs implémentées respectivement au dessus de Pastry et Chord.

3) L'application qui utilise la DHT : La couche applicative représentant toute application distribuée nécessitant l'utilisation d'une DHT. Tel est le cas du système de fichiers distribué Collaborative File System (CFS) [DKK⁺01].

- **Propriétés des Tables de hachage distribuées**

Les structures des DHTs sont influencées par les graphes (par exemple les arbres, les hyper-cubes, etc). Ces graphes présentent les propriétés ci dessous :

- Faible diamètre de routage : Le routage d'une requête s'exprime en $O(\log N)$ sauts dans la plupart des tables de hachage distribuées, avec N étant le nombre des nœuds dans le réseau. Dans un réseau à 2^{10} nœuds par exemple, le nombre de sauts est 10 pour une base binaire.

- Passage à l'échelle : La scalabilité est assurée grâce au fait du faible diamètre et de la constante taille ou logarithmique des tables de routage permettant aux DHTs une bonne propriété de passage à l'échelle.

- La tolérance aux pannes : Une meilleure robustesse face aux pannes est assurée par l'absence de centralisation. Des routes secondaires acheminent les requêtes de recherches, même en cas de churn. Des mécanismes de redondance sont souvent mis en place pour éviter la perte définitive ou l'inaccessibilité des ressources.

- L'équilibrage de charge : L'application de fonctions de hachage pour obtenir les identifiants des nœuds et les clés des objets contribue au partage équitable des données entre les nœuds. Cet équilibre implique l'équilibre du trafic dans

le cas où toutes les ressources sont demandées de manière équitable.

- **Topologies des DHTs**

-Topologie en anneau : La DHT Chord présente bien la topologie en anneau. Le déploiement de Chord est présent dans une multitude d'applications. Ainsi, la DHT est déployée dans le système de stockage de fichiers distribués CFS [DKK⁺01] et dans la version Pair-à-Pair du DNS DDNS [CMM02]. Les pairs dans la topologie de Chord sont placés dans l'anneau dans l'ordre croissant de leurs identifiants ID. Chaque clé est associée au nœud d'identifiant immédiatement supérieur. De ce fait, l'identifiant ID du pair est responsable de l'intervalle suivant des clés] prédécesseur(ID), ID].

- Topologie en arbre basée sur l'algorithme de Plaxton : Le deuxième groupe de tables de hachage distribuées repose sur l'algorithme de Plaxton [PRR97]. Dans ce groupe, nous trouvons Pastry, Tapestry, Bamboo [RGK⁺05], etc. Nous allons foculer dans le chapitre suivant sur Pastry parce que nous l'avons utilisé dans notre recherche.

-Topologie en hyper cube : CAN [RFH⁺01] représente une DHT avec une topologie en hyper cube. CAN propose un routage qui se déroule de proche en proche : à chaque saut, les coordonnées ne sont changées que sur une seule dimension. Néanmoins, nous trouvons d'autres structures utilisant la topologie en hypercube. Chord, peut être considéré parmi les hypers cube unidimensionnels ainsi, il repose sur un espace cartésien circulaire unidimensionnel. Kademlia [MM02] aussi représente cette topologie avec une dimension de moyen $\log(N)$, alors qu'elle est représentée souvent sous la forme d'un arbre binaire.

- **Tableau comparatif**

Parmi les travaux élaborées pour faire les études comparatives sur les performances théoriques des différentes tables de hachage distribuées nous citons [GGG⁺03] et [LKR03]. De plus, il y a eu des recherches dans [LSG⁺04] et [LNBK02] pour l'évaluation de la résistance des tables de hachage distribuées en présence du churn.

Le Tableau 1.1 récapitule les principaux projets de DHTs en livrant une analyse complète de leurs caractéristiques théoriques. Les propriétés principales

qui sont présentées sont le degré, le diamètre, et la flexibilité du routage et du voisinage. Le degré définit la taille de la table de routage. Le diamètre présente la distance maximale, en nombre de sauts, entre deux pairs. La flexibilité correspond au degré de liberté des nœuds lors la collection des nœuds à partir des tables de routage pour le prochain saut.

TAB. 1.1 – Performances des différentes tables de hachage distribuées

	Diamètre	Degré	Flex voisinage	Flex routage
Tapesty	$O(\log_{2^b} N)$	$O((2^{b-1}) * \log_{2^b} N)$	$N^{\frac{\log N}{2}}$	1
Pastry	$O(\log_{2^b} N)$	$O((2^{b-1}) * \log_{2^b} N)$	$N^{\frac{\log N}{2}}$	2^b
chord	$O(\log N)$	$O(\log N)$	$N^{\frac{\log N}{2}}$	$c(\log N)$
kademlia	$O(\log_{2^b} N)$	$O(K * \log_{2^b} N)$	$N^{\frac{\log N}{2}}$	2^b
Can	$O(d n^{\frac{1}{d}})$	$O(d)$	d	$c(\log N)$
viceroi	$O(\log N)$	$O(1)$	1	1

Nous notons :

b : le nombre de chiffres dans l'alphabet utilisé

N : le nombre de nœuds

k : le nombre de voisins dans chaque sous-arbre d'un nœud (pour Kademlia).

d : le nombre de dimensions avec $d = 2b$ (pour CAN).

D'une première vue nous remarquons la similitude des propriétés entre les DHTs. Précisément, le nombre de sauts moyen pour le routage d'une requête s'exprime souvent en fonction de $O(\log N)$. La principale différence entre les tables de hachage distribuées figure dans le degré de connectivité et le degré de flexibilité. Pour le degré de connectivité, nous distinguons deux types de connectivités : logarithmique et constant. L'aspect logarithmique garantit le passage à l'échelle ce qui constitue une caractéristique importante pour une exploitation dans un environnement Pair-à-Pair à large échelle. L'aspect constant est très bénéfique pour deux raisons : la première est le passage à l'échelle en assurant l'aspect constant dans la taille de la table de routage indépendamment de la taille du réseau. La deuxième raison est la maîtrise des coûts de la maintenance des pairs du réseau, car la mise à jour des entrées de la table de routage dépend du nombre de nœuds voisins. Cette propriété de performance est intéressante pour les réseaux d'énorme taille. Concernant la

propriété de flexibilité, nous constatons que les tables de hachage distribuées à degré constant sont moins flexibles. Remarquons que Chord est la DHT la plus flexible. CAN est caractérisée par une flexibilité dans le routage mais pas dans le choix du voisinage, au contraire de Pastry, Tapestry et Kademia. Enfin, Viceroy représente la DHT la moins flexible et la moins résistante en présence des pannes [GGG⁺03].

La plupart de ces projets DHT ont été effectués entre 2001 et 2003. De nos jours, le domaine de recherche dans les DHTs s'élargit et plusieurs travaux d'optimisations ont été élaborés pour l'amélioration des systèmes Pair-à-Pair basés sur les DHTs.

1.4 Conclusion

Dans la première partie de ce chapitre, nous avons présenté un récapitulatif sur les systèmes Publier/Souscrire en détaillant leurs caractéristiques et leur modèle de communication. Ces discussions nous permettent d'arriver à conclure que le modèle de communication Publier/Souscrire est un bon candidat pour les applications temps réel et à grande échelle. En effet, la nature de la communication dans le modèle Publier/Souscrire est anonyme et asynchrone. Les fournisseurs et les consommateurs d'information sont totalement découplés. Ceci fournit trois effets importants. Premièrement, un participant peut fonctionner dans un système sans se rendre compte de l'existence d'autres composants. La seule connaissance qu'il faut connaître est la structure des événements notifiés auxquels il s'intéresse. Deuxièmement, l'ajout et la suppression d'un composant dans le système ne cause pas beaucoup d'influence sur les autres composants. Troisièmement, le modèle de communication et de coordination de Publier/Souscrire convient mieux que le modèle client/serveur traditionnel à faire face à la déconnexion non-annoncée des composants, ce qui caractérise les réseaux à forte dynamique. De plus, les mises à jour sont réalisées automatiquement. Les souscripteurs ne cherchent pas les nouveaux événements eux-mêmes, ils ne reçoivent que ceux qui les intéressent, et seulement les nouvelles publications.

Dans la deuxième partie de ce chapitre, nous avons présenté une vue d'ensemble

sur les réseaux Pair-à-Pair. Une classification de ces réseaux permet d'en extraire deux catégories qui sont les non structurés et les structurés. Parmi les réseaux Pair-à-Pair structurés distribués, nous avons évoqué les DHTs. Comme nous avons vu, différentes topologies ont été présentées par les DHTs. Malgré que ces DHTs présentent des propriétés pertinentes et ont solutionné plusieurs problèmes mais ils présentent encore certaines limites comme faire face à la forte dynamique des nœuds. En effet, l'arrivée et le départ fréquents et imprévisibles des pairs dans le réseau aussi bien que les défaillances entraînent des recherches non aboutissantes à cause de la perte des données. Ainsi, cette limite a présenté un défi pour les concepteurs de tables de hachage distribuées pour que ces systèmes résistent aux changements perpétuels des nœuds du réseau et garantissent la disponibilité des données dans le réseau. Dans le chapitre suivant, nous exposons des travaux proposés, étudiant le comportement des DHTs face au churn.

Chapitre 2

Systemes publier/souscrire basés DHT & dynamisme

2.1 Introduction

En pratique, la forte dynamicité du réseau Pair-à-Pair due à l'arrivée et au départ non anticipés des pairs forme une limite pour la disponibilité et la performance des systèmes Publier/Souscrire à base de DHT. De telles modifications conduisent au pire des cas à la perte des tables de routage des souscriptions, et au meilleur des cas à une dégradation de la performance du système. Ainsi, nous allons passer en revue quelques traitements de la dynamicité dans les arbres, les DHTs et les systèmes Publier/Souscrire.

2.2 Dynamicité des arbres

D'après [CF05], une approche a été proposée pour la reconfiguration de la topologie en exploitant une DHT. L'algorithme prend en charge la topologie des arbres et traite très bien avec la dynamicité du réseau en limitant l'impact de la reconfiguration induite par les changements de la topologie. L'algorithme exploite une table de hachage distribuée pour cartographier les clients sur la topologie de l'arbre. Les résultats de simulation valident l'efficacité de l'approche à la fois dans le contrôle du nombre de voisins des hôtes dans le réseau et dans la minimisation de l'impact

de l'algorithme sur le protocole responsable de la reconfiguration de la couche de routage. L'algorithme prend une structure prédéfinie d'arbre et mappe chaque hôte participant aux nœuds correspondants en exploitant la DHT. Ainsi, à chaque changement, l'algorithme permet de mettre à jour la liste des parents et des nœuds feuilles et de contrôler les changements dans la liste des voisins.

Autres études ont été élaborées dans [FM08] qui consistent à présenter un nouveau protocole qui, directement et efficacement maintient la structure des arbres en présence du churn. En effet, ce protocole est la combinaison des techniques des récupérations des parents lors de leur perte. Les nœuds surveillent l'état de leurs voisins par le suivi du trafic au niveau applicatif ou en envoyant des messages de vérification périodiquement. Dès qu'un nœud détecte l'échec de son parent dans l'arbre (étape 1), il identifie activement un nouveau parent candidat (étape 2). Quand il localise un, une PARENTREQUEST est envoyée (étape 3). Puis le candidat vérifie s'il peut accepter la demande en toute sécurité. Si c'est le cas, le processus se termine avec l'arbre reconnecté, sinon le processus se répète. Dans des cas extrêmes, un nouveau parent ne sera pas trouvé, et le processus se termine par le nœud se déclarant lui-même la racine. Ainsi, la sélection des candidats parents doit être effectuée avec des changements bien localisés.

2.3 Exemples DHT et Publier/Souscrire

2.3.1 La DHT Pastry

Pastry est un réseau de recouvrement et de routage pour la mise en œuvre d'une DHT similaire à Chord. Les paires (clé-valeur) sont stockées dans un réseau redondant Pair-à-Pair d'hôtes connectés à l'internet. Le protocole est amorcé en lui fournissant l'adresse IP d'un pair déjà dans le réseau puis la construction et la réparation du pair se fait dynamiquement dans la table de routage. En raison de sa nature décentralisée et redondante, il n'y a pas de point de défaillance unique et tout nœud peut quitter le réseau à tout moment sans préavis. Le protocole est également capable d'utiliser une métrique de routage fournie par un programme à l'extérieur,

telles que ping ou traceroute, afin de déterminer les meilleures routes pour le stockage dans sa table de routage.

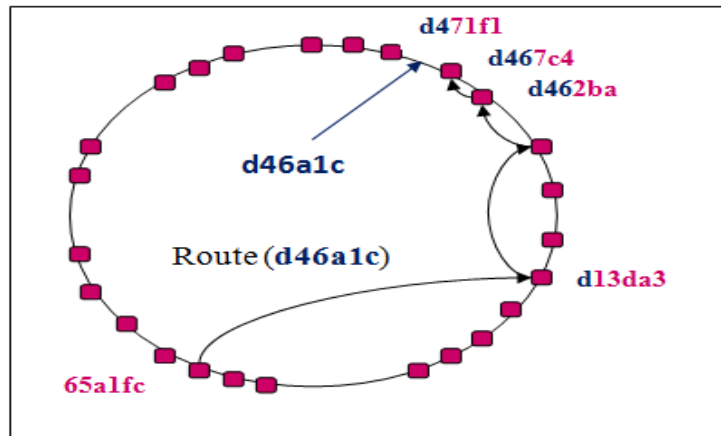


FIG. 2.1 – Routage vers la clé (d46a1c)

Dans Pastry, chaque pair ou objet se voit attribuer un identifiant (un `nodeId` ou `objectId`) de 128 bit. Un identifiant de nœud est utilisé pour positionner le nœud dans un espace circulaire d'identifiants, qui va de 0 à $2^{128} - 1$. Les identifiants des nœuds sont choisis aléatoirement et de manière uniforme donc les pairs qui sont adjacents à l'identifiant du nœud sont géographiquement diversifiés. Chaque nœud est responsable d'une fraction de l'espace identifiant. Pour insérer une valeur, nous commençons par le hachage de cette valeur pour retrouver la clé (identifiant). Ensuite, nous attribuons cette clé à l'intervalle spécifique d'identifiant. Enfin, cette valeur est stockée sur le nœud correspondant à cette gamme d'identifiant. Ainsi, Pastry stocke un objet dans un nœud dont l'identifiant est le plus proche numériquement à l'identifiant de l'objet comme le montre la figure 2.1.

Pair 10233102			
Voisins virtuels			
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232
Table de routage			
-0-2212102	1	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	2	10-3-23302
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
		2	
Voisins réels			
13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

FIG. 2.2 – Table de routage du pair 10233102

Le routage dans Pastry est hybride : au début un routage en préfixe suivant l’algorithme de Plaxton, puis à la fin, un routage selon la topologie en anneau similaire à Chord. Le protocole Pastry représente la base de PAST [ID01] (système de stockage de données anonyme) et de la diffusion dans Scribe .

La couche de routage est formée au dessus de la table de hachage par chaque pair découvrant et échangeant des informations d’état composé d’une liste de nœuds feuilles, une liste de voisinage, et une table de routage comme illustré dans la figure 2.2. La liste des nœuds feuilles se compose des $L/2$ les plus proches pairs à l’identifiant du nœud dans chaque direction autour du cercle. En plus des nœuds feuilles, il ya aussi la liste de voisinage. Cette liste contient les M pairs les plus proches en terme de la métrique de routage. Même si elle n’est pas utilisée directement dans l’algorithme de routage, la liste de voisinage est utilisée pour maintenir les principes de la localité de la table de routage. En outre, les entrées à la ligne i de la table de routage partagent i premiers chiffres avec l’identifiant du nœud.

Un paquet peut être acheminé à n’importe quelle adresse dans l’espace des clés s’il ya un pair avec cet identifiant du nœud ou non. Le paquet est routé et le nœud dont l’identifiant est le plus proche de la destination souhaitée recevra le paquet. Chaque fois qu’un pair reçoit un paquet ou veut envoyer un paquet, il examine d’abord ses nœuds feuilles puis il route directement au nœud correct s’il le trouve. Si cela échoue, le pair consulte sa table de routage dans le but de trouver l’adresse

d'un nœud qui partage un plus long préfixe avec l'adresse de destination que le pair. Si le pair n'a pas de contacts avec un plus long préfixe ou le contact est mort, il va choisir un pair de sa liste de voisins réels avec la même longueur de préfixe que lui mais numériquement plus proche de la destination et lui envoie le paquet. Ainsi le nombre de chiffres corrects avec l'adresse désirée augmente ou reste le même et s'il reste le même la distance entre le paquet et sa destination devient plus petite donc le protocole de routage converge, ceci est illustré dans la figure 2.2.

De plus, Pastry est caractérisé par son support de la dynamique du réseau. En fait, l'arrivée et le départ des nœuds dans le réseau Pastry se déroule de cette manière. Ainsi, un nouveau nœud qui vient de rejoindre le réseau doit initialiser ses tables d'état (la table de routage et de l'ensemble des feuilles), puis informe les autres de sa présence. Supposons que X est l'identifiant du nouveau nœud. Il repère d'abord un nœud dans le réseau en utilisant par exemple "l'expansion anneau IP multi-cast". Supposons que l'identifiant du nœud trouvé est A. Maintenant le nœud X envoie un message de jointure spécial avec la clé X au nœud A. Comme pour les messages normaux, le nœud A route ce message au nœud Z qui est le plus proche numériquement à X. Chaque nœud du chemin vers le nœud Z envoie une rangée de sa table de routage au nœud X. Le nième nœud envoie son nième rangée. Le nœud Z envoie son leaf set au nœud X. Finalement, le nœud X informe n'importe quel nœud qui a besoin d'être informé de son arrivée. Comme les nœuds peuvent tomber en panne ou quitter sans avertissement, des nœuds voisins dans l'espace des clés (par exemple, les nœuds dans le leafset) échangeront périodiquement des messages keep-alive. Si un nœud ne répond pas pendant un temps T, il est considéré comme échoué et tous les membres de son leafset mettent à jour leurs leaf set.

2.3.2 Le système Publier/Souscrire Scribe

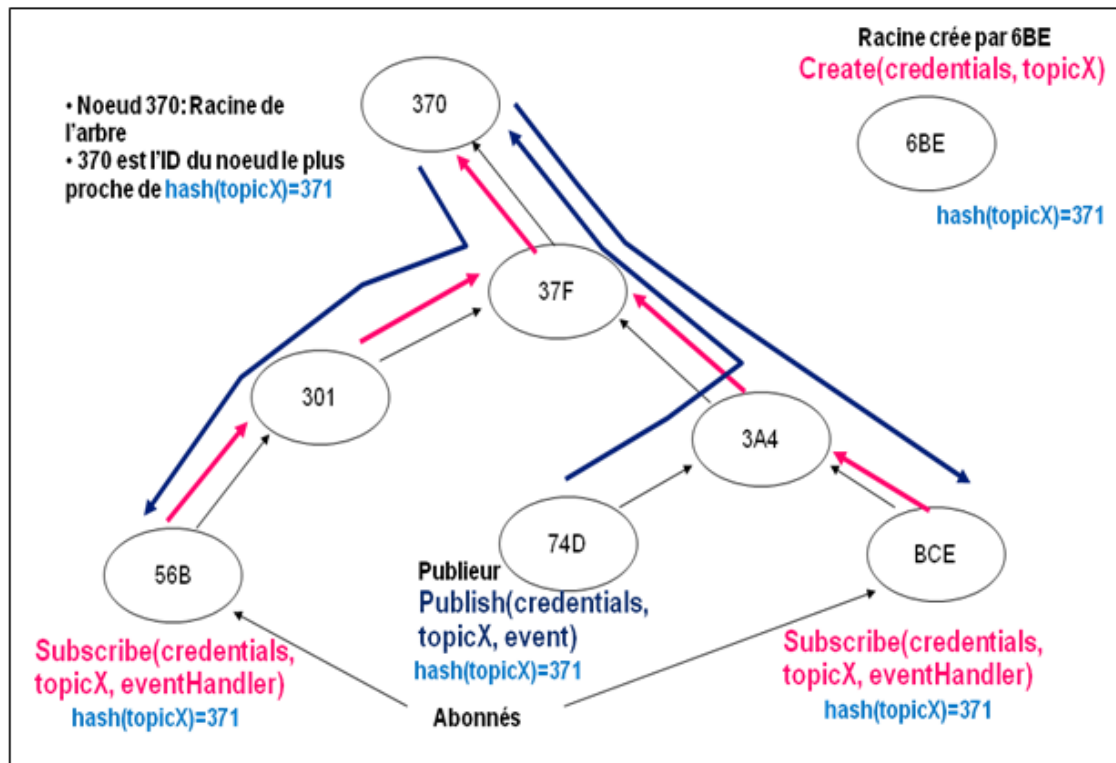


FIG. 2.3 – Arbre de Scribe

Scribe est un système Publier/Souscrire décentralisé basé sujet qui utilise Pastry dans le routage et la recherche des clés. Les utilisateurs créent des sujets auxquels les autres utilisateurs peuvent s'abonner. Une fois que le sujet a été créé, le propriétaire du sujet peut publier de nouvelles entrées dans le cadre du sujet qui sera distribué dans un arbre multicast pour tous les nœuds Scribe qui ont souscrit à ce sujet. Le système fonctionne en calculant la valeur de hachage du nom du sujet. Ce hachage est ensuite utilisé comme une clé de Pastry, puis les paquets sont routés au plus proche nœud à la clé en utilisant le protocole de routage Pastry pour créer le nœud racine de ce sujet sur ce nœud. Les souscripteurs ensuite s'abonnent à ce sujet par le calcul de la clé du sujet, puis utilisent Pastry pour acheminer un message vers le nœud racine. Lorsque le nœud racine reçoit le message de la souscription à partir d'un autre nœud il ajoute l'identifiant du nœud à la liste de ses enfants et commence à agir en tant que transitaire du sujet. Ainsi, Scribe construit un arbre

de multidiffusion enraciné à chaque nœud racine de sujet comme suit : Quand une souscription trouve sa route à la racine du sujet, tous les nœuds intermédiaires le long de la route sont ajoutés à l'arbre de notification multicast s'ils n'appartiennent pas déjà à l'arbre. Cette optimisation rend la souscription efficace et complètement distribuée. Pour le multicast, la publication est envoyée à la racine du sujet, à partir duquel la publication est envoyée tout le long de l'arbre multicast.

La décentralisation est accomplie par le fait que tous les nœuds du réseau fouillent sur les messages de souscription en passant devant eux sur leur chemin vers le nœud racine. Si le sujet est celui auquel le nœud courant souscrit, il cessera de transférer le paquet vers le nœud racine et ajoute le nœud souscripteur à ses enfants. De cette façon, une structure arborescente est formée par le nœud racine au sommet comme l'illustre l'arbre de la figure 2.3. Lors de la réception d'une publication, le nœud racine l'envoie aux premiers nœuds souscripteurs puis chacun de ces nœuds transmet les messages à ses enfants, et ainsi de suite.

2.4 Notion du churn

Le mot churn (contraction de l'anglais change and turn) symbolise le rapport des arrivées et des départs de nœuds durant une période de temps. Ainsi, nous utilisons le terme churn pour désigner le dynamisme des nœuds.

2.5 Problème du churn

Une DHT peut échouer pour compléter les requêtes de recherche, ou elle peut les compléter, mais les résultats sont incompatibles pour la même recherche lancée par différents nœuds sources. L'incohérence est due au départ des nœuds qui mémorisent une donnée à chercher. En effet, le dynamisme des nœuds mène à une reconfiguration perpétuelle des nœuds du réseau. D'autre part, une DHT peut continuer à retourner des résultats cohérents alors que le taux de churn augmente, mais elle peut souffrir d'une augmentation de la latence dans le processus de la recherche. De tels problèmes ont présenté des champs de recherche pour les concepteurs des

DHTs.

2.6 Effet du churn sur les performances des DHTs

D'après [LMSM09], le réseau Pair-à-Pair est en face de plusieurs changements sous l'effet d'un taux de churn élevé, et le protocole de maintenance est obligé de migrer fréquemment des blocs de données. Alors que certaines migrations sont nécessaires pour restaurer k copies de données, d'autres pour maintenir les nœuds invariants. Comme exemple de scénario nous considérons un nouveau pair qui se connecte au système possédant un identifiant plus proche du bloc de données que la racine donc il devient la nouvelle racine. Dans ce cas, le bloc de données migre sur le pair arrivant. Ce phénomène se produit aussi dans la réplication à base de leafset surtout que l'extension de la taille du replica-set augmente la probabilité pour un nouvel arrivant de provoquer des migrations de blocs.

Dans la réplication à base de clés multiples, un nouvel arrivant peut être inséré entre deux réplicas sans migration de blocs de données à condition de ne pas devenir une nouvelle racine d'un bloc. Mais l'inconvénient est que la maintenance doit être effectuée pour chaque bloc d'une façon séparée ce qui ne permet pas le passage à l'échelle en termes de blocs par pair.

2.7 Traitement du churn dans la DHT

2.7.1 La Réplication dans les DHTs

D'après [LMSM09] il existe deux protocoles qui gèrent les réplicas : le protocole de placement initial et le protocole de maintenance. Les protocoles de placement de réplicas présentent deux principales stratégies : basée sur le voisinage (leafset), et basée sur les clés multiples :

2.7.1.1 Protocole de placement Initial

- Stratégie basée sur le voisinage

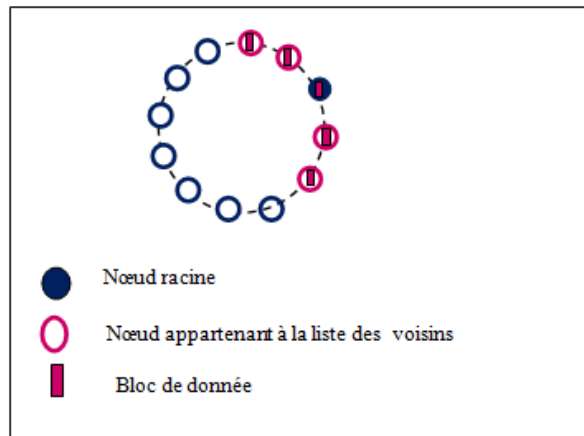


FIG. 2.4 – Réplication basée sur le voisinage

La racine du bloc de données stocke une copie du bloc. Le bloc est également répliqué sur un sous-ensemble du leafset (les voisins les plus proches) de la racine. Ces répliques peuvent être soit prédécesseurs, soit successeurs de la racine dans l’anneau, soit les deux. Ainsi, les copies d’un bloc sont stockées de manière contiguë dans l’anneau, comme cela est illustré par la Figure 2.4. Cette approche a été implémentée dans DHash [DLS⁺04]. La réplication par successeur est une variante de la stratégie précédente, dans laquelle les données sont répliquées sur les successeurs immédiats de la racine au lieu de l’être sur les pairs les plus proches.

Toutefois, Cette stratégie présente des limites, ainsi elle exige de fortes contraintes de placement des répliques ce qui induit à un gaspillage de bande passante en cas de churn.

- **Stratégie basée sur les clés multiples**

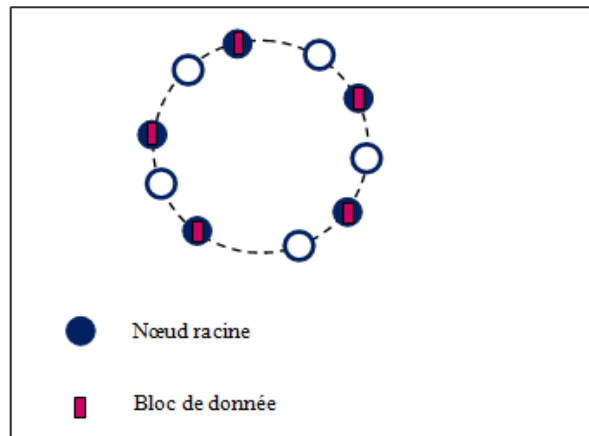


FIG. 2.5 – Réplication à base de clés multiples

Cette stratégie illustrée par la figure 2.5 présente le calcul de K clés de stockage différentes pour tout bloc de données, chaque clé correspondant à une racine distincte du bloc. Les blocs de données sont ensuite répliqués sur ces K pairs racines. Cette approche a été implémentée dans CAN [RFH⁺01].

Toutefois cette stratégie nécessite un coût élevé en nombre de messages si beaucoup de blocs par nœud et donc présente des difficultés de passage à l'échelle.

2.7.1.2 Protocole de maintenance

Les protocoles de maintenance conservent K copies de chaque bloc de données en respectant l'approche de placement initiale. Cela signifie que les K copies de chaque donnée doivent être stockées sur les K voisins contigus du pair responsable dans la stratégie basée sur le voisinage et sur les K racines dans la stratégie basée sur les clés multiples. Le mécanisme de maintenance à base de leafset fait un échange périodique d'information au sein des leafsets. Dans le cas des stratégies de réplication à base de clés multiples, la maintenance doit être effectuée séparément pour chaque bloc de données. Ainsi, il est nécessaire de vérifier périodiquement pour chaque bloc de données stocké dans le système si les différentes pairs racines sont encore en vie et stockent encore une copie du bloc de données.

2.7.2 Relaxation des contraintes de placement de la DHT pour tolérer le churn

RelaxDHT [LMSM09] est construite au dessus d'une couche de routage basée clé telle que Pastry ou Chord permet d'éviter de copier des blocs de données quand ce n'est pas nécessaire à la restauration d'un réplica manquant. Cette solution repose sur la réplication à base de leafset qui relâche les contraintes de placement dans le leafset. Cette solution réduit significativement le nombre de transferts de blocs de données lorsque des pairs rejoignent ou quittent le système. Ce qui permet au mécanisme de maintenance de mieux tolérer le churn. Il introduit cependant quelques effets de bord. Ainsi, la distribution des blocs de données résultante est biaisée. En présence de churn, les pairs les plus anciens stockent plus de blocs de données que les nouveaux pairs car les transferts ne sont effectués que dans les cas nécessaires. De plus, avec RelaxDHT, il nous vient d'avoir certaines racines de blocs qui présentent des incohérences temporaires, causant un surcoût de communication pendant la phase de la recherche de la donnée. Par exemple, il nous vient souvent que la panne d'un pair racine engendre la non obtention du bloc de données malgré l'existence des copies du bloc correspondant dans le système car le nouveau pair dont l'identifiant est le plus proche peut ne pas connaître le bloc. Cela reste vrai jusqu'à ce que la panne soit détectée par un des pairs du replica-set et réparée.

2.8 Conclusion

Nous avons combiné dans ce chapitre la notion du churn et des systèmes publier/souscrire basés DHT. Dans un premier temps, nous avons cité quelques systèmes existants qui présentent le paradigme Publier/Souscrire basé DHT. Dans un deuxième temps, nous avons introduit le problème du churn et son effet sur les performances des DHTs et nous avons détaillé quelques travaux d'améliorations proposées dans le cadre des structures des tables de hachage distribuées, notamment l'exploitation de la réplication pour optimiser la disponibilité des données dans le réseau face au churn.

De ce point de recherche, nous proposons une approche de résistance au churn pour les systèmes Publier/Souscrire basés DHT. Notre approche vise à maintenir un niveau de matching acceptable entre les souscriptions et les publications diminuant ainsi le taux de perte des tables de routage des souscriptions lors de l'application de différents niveaux de churn.

Chapitre 3

Conception et évaluation de l'approche proposée

3.1 Introduction

Le but principal dans la gestion des données dans les réseaux Pair-à-Pair est de garder les données toujours disponibles. Pour s'accomplir, nous élaborons une nouvelle stratégie de référencement par digit complémentaire en s'inspirant du principe de P-Grid. Notre approche se base sur la méthode de référencement par prévention pour assurer la disponibilité des souscriptions dans un système Publier/Souscrire basé DHT dynamique en minimisant le maximum le trafic des messages.

Dans la suite, nous effectuons la description détaillée de notre approche de référencement, aussi nous présentons un ensemble de tests pour l'évaluation de notre approche. Ce chapitre sera clôturé par l'interprétation des résultats expérimentaux.

3.2 Approche proposée

Nous avons choisi de travailler sur le système publier/souscrire basé sujet Scribe en perspective le basé contenu P2P-TOPSS. Ainsi ce système a été choisi pour sa structure résistante à l'arrivée et au départ perpétuels des nœuds.

3.2.1 Justification du choix du système

3.2.1.1 Pastry

Nous avons choisi Pastry vu qu'il s'agit d'un réseau de recouvrement scalable, décentralisé et auto-organisable qui s'adapte automatiquement à l'arrivée et au départ continuels des nœuds. De plus, Pastry utilise une fonction de hachage consistante pour attribuer les données aux nœuds ce qui permet d'éviter les collisions et d'équilibrer la charge au sein du réseau.

3.2.1.2 Scribe

L'initiative du choix de Scribe a été élaborée vu qu'il est un système Publier/-Souscrire et qu'il est structuré au dessus de la table de hachage distribuée Pastry. De cette façon, il permet bien le passage à l'échelle, l'évolutivité du système et le dynamisme du réseau.

3.2.1.3 P2P-TOPSS

Nous avons utilisé le code de P2P-TOPSS acronyme de (peer-to-peer-based Toronto Publish/Subscribe System) vu qu'il est une implémentation de Scribe sous java et qu'il est extensible en basé contenu.

3.2.2 Concept P-Grid

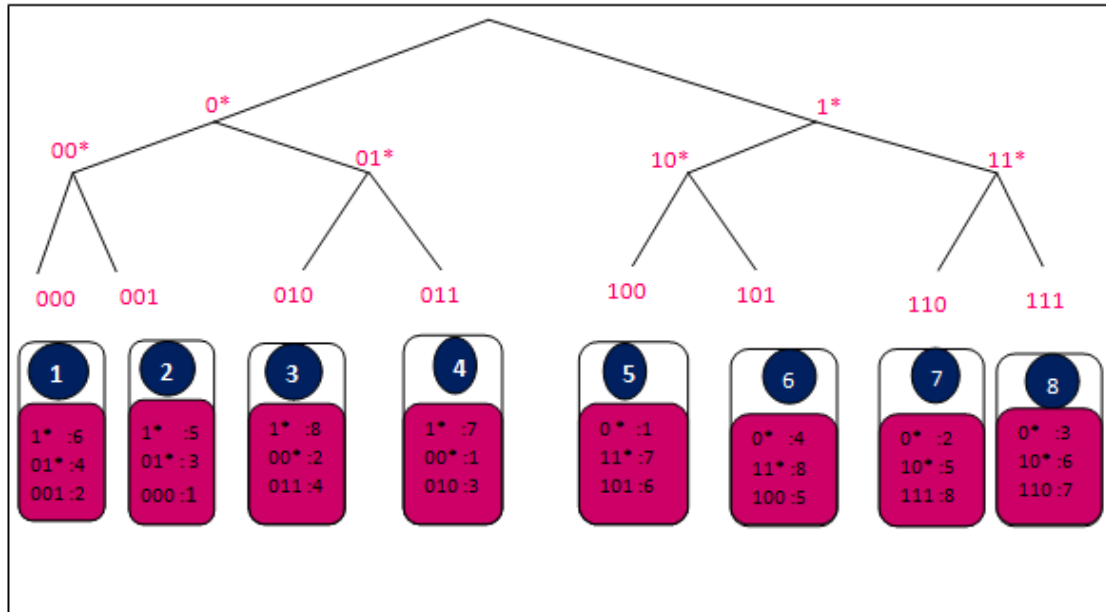


FIG. 3.1 – Arbre de P-Grid

P-Grid [ACMD⁺03] est un système Pair-à-Pair structuré qui forme un arbre virtuel dont le chemin de la racine à chaque feuille indique l'ensemble des clés gérées par un pair (la représentation binaire du chemin est un préfixe de clé). Ainsi, Chaque pair gère toutes les clés qui partagent un préfixe identique à ce chemin. Ainsi, P-Grid est un arbre binaire structuré au dessus du réseau physique Internet utilisant le routage basé préfixe pour assurer des recherches efficaces.

D'après la figure 3.1 plusieurs pairs sont associées à la même partition de l'espace clé (réplication de structure), et les pairs maintiennent de multiples références à des pairs avec le même chemin (la réplication des données). Ainsi, l'espace clé est divisé en partitions définies par les chaînes binaires.

Pour la recherche, chaque pair maintient des références à d'autres pairs / partitions à chaque niveau de l'arbre. Pour chaque bit de son identifiant chaque pair stocke l'adresse d'au moins un pair qui est responsable de l'autre côté de l'arbre binaire à ce niveau cela signifie que l'identifiant de la référence contient le complémentaire du bit de l'identifiant du pair à ce niveau. De cette façon, l'algorithme de la construction

de P-Grid garantit que les chemins vers les clés demandées sont toujours fournis et que les requêtes sont toujours satisfaites.

3.2.3 Spécification de notre approche de référencement

Pastry figure parmi les protocoles les plus populaires dans les réseaux DHT, dans lequel les études sont approfondies dans les services de stockage et de localisation des données au niveau du réseau. Toutefois, lors de l'élévation de l'intensité du dynamisme, les notifications risquent de ne pas arriver aux souscriptions souhaitées à cause de la perte des tables de routage des souscriptions. Pour résoudre ce problème, nous suggérons un référencement au niveau de la mise des souscriptions pour assurer la disponibilité des souscriptions dans le service d'événement. Notre approche est appliquée sur le système Publier/Souscrire Scribe structuré au dessus de Pastry présenté dans le deuxième chapitre de ce rapport.

Ainsi, notre but est d'assurer la disponibilité de souscriptions dans le service d'événements, même à un niveau de dynamisme très fort, en réservant le nombre de références nécessaires. Le taux de référencement est paramétrable suivant le taux des changements dans la topologie du réseau et donc nous augmentons le nombre de références pour un taux faible de stabilité des nœuds dans le service d'événements.

3.2.3.1 Phase de souscription

Un souscripteur se connecte au service d'événements pour spécifier les contraintes qu'il désire avoir sur les attributs. Ensuite cette contrainte est hachée par la fonction de hachage de pastry et s'est translaturée à une clé hachée dans la DHT. La souscription est alors envoyée au nœud possédant l'identifiant le plus proche de la clé de la souscription. Quand une souscription trouve sa route à la racine du sujet, tous les nœuds intermédiaires le long de la route sont ajoutés à l'arbre de notification multicast s'ils n'y appartiennent pas. Puis, la procédure de référencement commence à s'exécuter. Ainsi, les références sont choisies selon leur position par rapport au nœud d'origine de façon qu'ils touchent des positions de différents niveaux de distance relativement au nœud d'origine. Nous nous sommes inspirés du principe de

P-Grid, ainsi nos calculs sont faits sur les complémentaires des digits pour obtenir les références souhaitées. La formule du complémentaire est $(d+8)\% 16$ avec d le digit concerné. Nous avons adopté cette méthode de référencement pour que la souscription ait des références qui acheminent différents chemins lors de leurs parcours vers la destination avec un choix précis de ces chemins pour minimiser le trafic des messages. Ce qui contribue à une différenciation au niveau des nœuds intermédiaires et donc à une augmentation du taux de disponibilité de la souscription dans le réseau ce qui assure une résistance efficace face à une topologie très dynamique des nœuds du réseau avec le minimum de trafic.

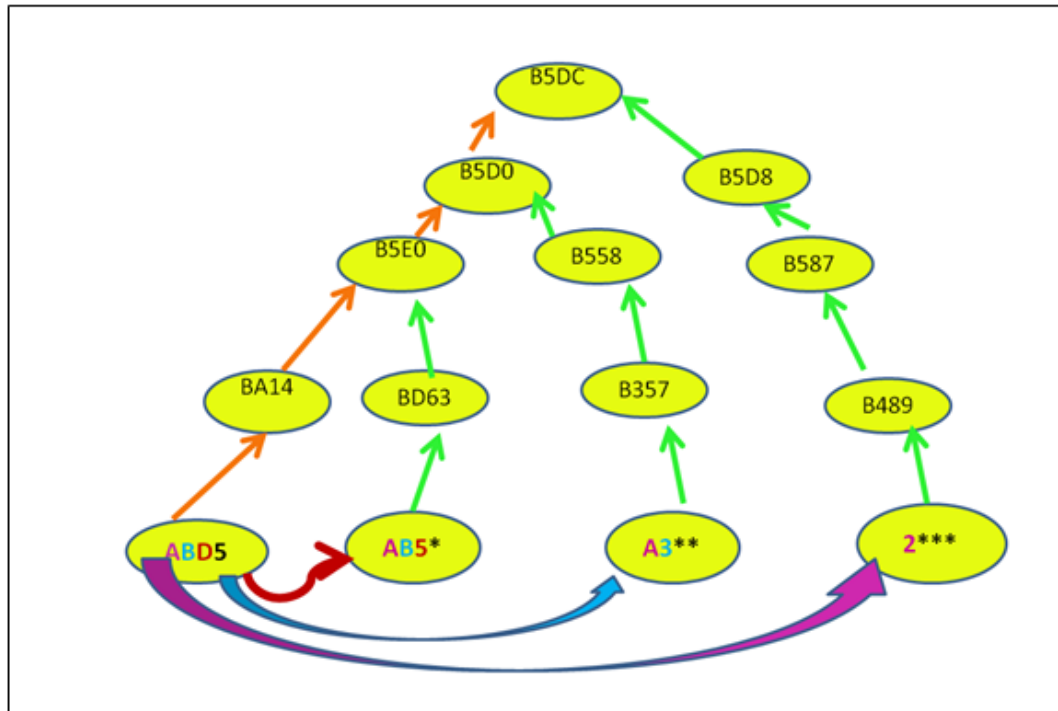


FIG. 3.2 – Exemple de référencement

La figure 3.2 illustre un exemple de référencement. Ainsi, l'identifiant du nœud d'origine est "ABD5". De ce fait, l'application de notre formule sur cet identifiant pour le premier digit va nous tomber sur un nœud débutant par le digit "2". Ce qui nous conduit à l'obtention d'un chemin vers la racine du sujet totalement différent. Pour le calcul suivant nous allons maintenir le premier digit c'est-à-dire la référence

va avoir le même premier digit "A" en faisant le calcul sur le deuxième digit. Ainsi nous obtenons une référence avec le préfixe "A3". Pour le troisième calcul nous allons conserver le préfixe "AB" pour la référence et faire le complémentaire sur le troisième digit. Et ainsi de suite.

Dans le cas de la perte de la racine nous nous contentons de la méthode de base de scribe qui consiste à chercher de nouveau une nouvelle racine.

Voici un pseudo code simplifié du fonctionnement de notre approche :

Listing 3.1 – **Procedure** souscription (nodeNum :Entier,tid :NodeId, sub :Subscription)

```
debut
  //creation du sujet
  procedure creer (nodeNum :Entier , tid :NodeId)
  //joint de l'arbre
  procedure joindre (nodeNum :Entier , tid :NodeId , sub :Subscription)
  //calcul de la premiere reference
  procedure Calculref1 ( nodeNum :Entier)

  //lancement de la souscription dans la reference
  procedure souscriptionref ( nodeNumref1 : Entier)

  procedureCalculref2 ( nodeNum :Entier)
  procedure souscriptionref (nodeNumref2 : Entier)

  //de meme pour les autres references
fin
```

Listing 3.2 – **Procedure** Calculref1 (nodeNum :Entier)

```
complementaire_premier , premier_digit :Entier , id : NodeId
debut
  //Extraire l'identificateur
  id=retourner_identificateur (nodeNum)
```

```
//Extraire le premier digit
premier_digit=Premierdigit(id)
//calculer le complementaire
complementaire_premier=( premier_digit +8)%16
// rechercher les references
pour n=1 a 1000 faire
    si (Premierdigit(neud_n)==complementaire_premier) alors
        ajouterauxreferences(neud_n)
    finsi
finpour
//trier les references selon la distance minimale
// par rapportà l'origine
pour i=1 a nombre_references faire
    select_min(neud_i)
fin pour
fin
```

Listing 3.3 – **Procedure** souscriptionref (nodeNum :Entier ,tid :NodeId, sub :Sub-
scription)

```
debut
    procedure creer ( nodeNum : Entier ,tid:NodeId)
    procedure joindre (nodeNum : Entier ,tid:NodeId ,sub:Subscription)
fin
```

3.2.3.2 Phase de publication

Lors d'un multicast d'une publication, l'acheminement de la notification vers la souscription correspondante peut être assuré à travers les différents chemins des références ajoutées même face à un dynamisme très fort des nœuds tout en minimisant le trafic des messages. Ainsi comme l'illustre les figures 3.3, 3.4 , 3.5 si une branche est perdue alors le matching est garanti par les autres chemins référencés.

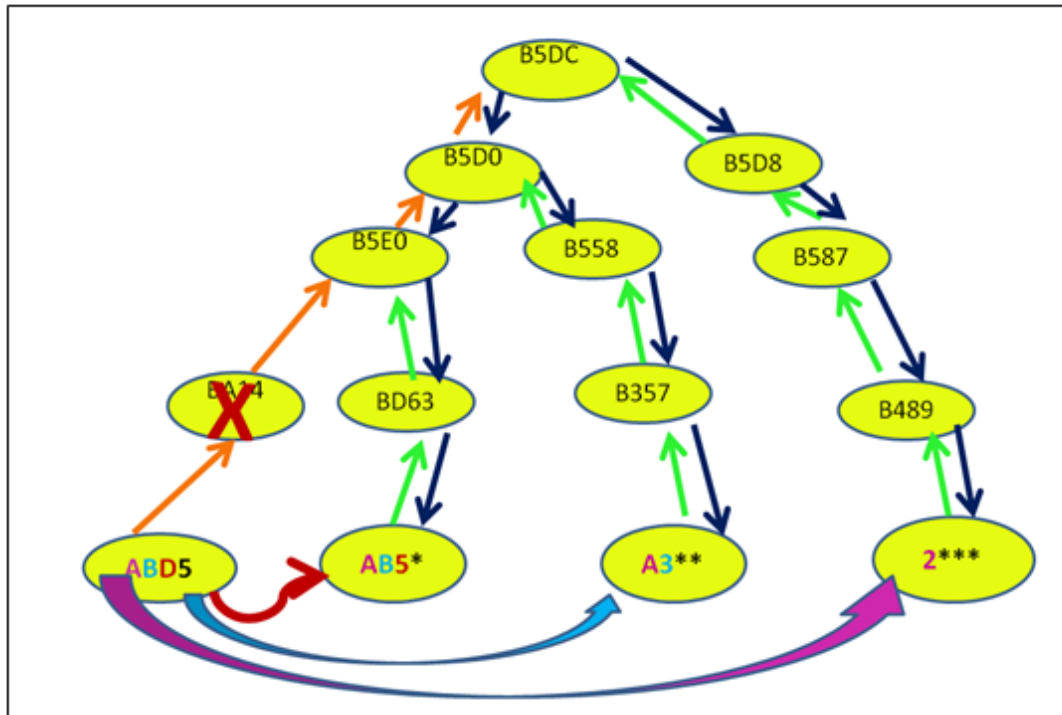


FIG. 3.3 – Exemple 1 de publication

La figure 3.3 illustre un scénario où le nœud "BA14" est affecté, donc lors du multicast de la publication le matching est assuré à travers les chemins ne contenant pas le nœud affecté. Dans ce cas il existe trois alternatives pour faire le matching.

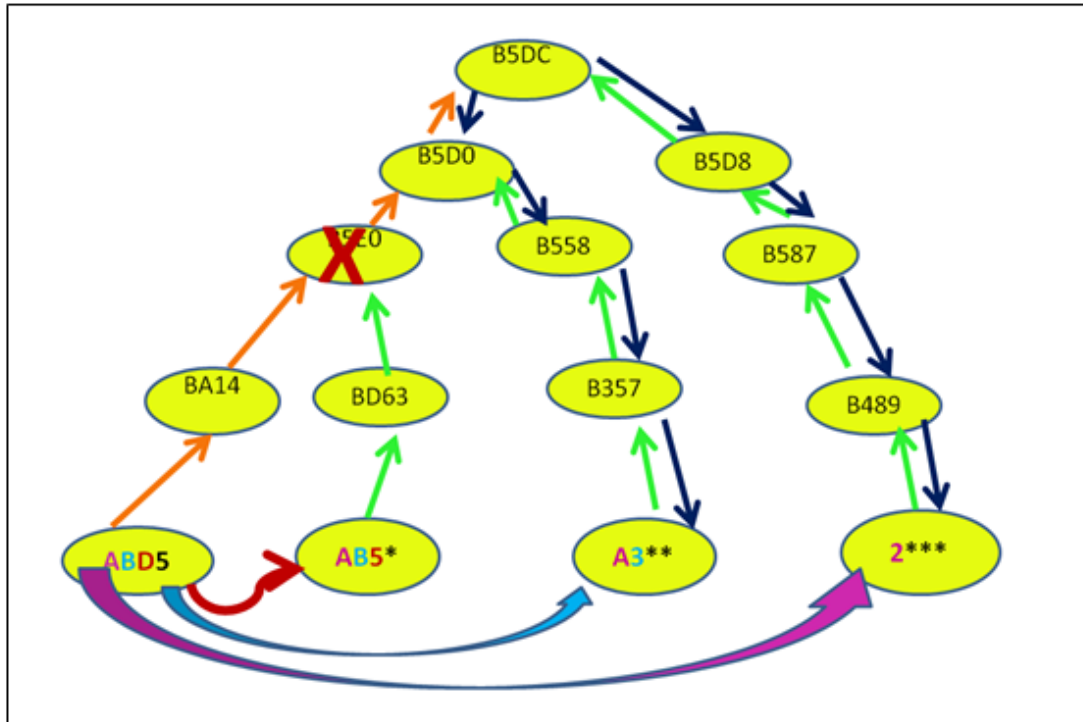


FIG. 3.4 – Exemple 2 de publication

La figure 3.4 illustre le scénario où le nœud affecté est plus proche de la racine. Donc lors du multicast de la publication le matching se déroule avec les références les plus loins.

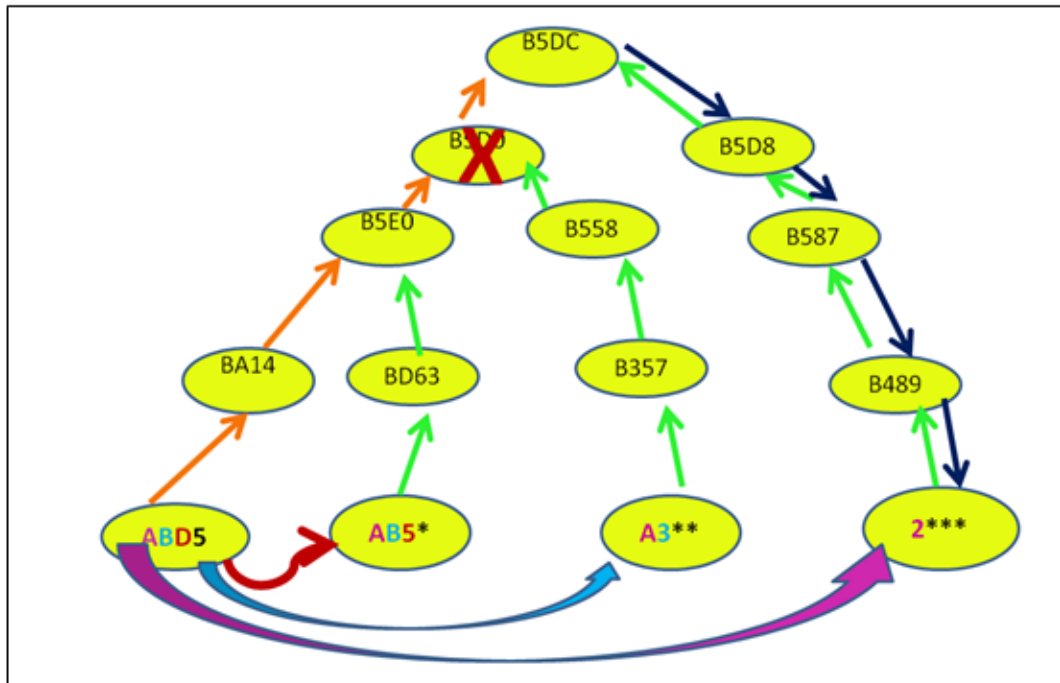


FIG. 3.5 – Exemple 3 de publication

La figure 3.5 illustre l'affectation du nœud le plus proche de la racine donc le matching est assuré par la référence ayant un chemin totalement différent par rapport aux autres.

3.3 Implémentation et Evaluation

Dans la suite, nous présentons l'environnement logiciel que nous avons utilisé pour mettre en place l'approche proposée. Nous détaillons par la suite le scénario de tests effectués ainsi que les résultats obtenus.

3.3.1 FreePastry

Les stratégies de résistance au churn doivent être testées afin d'estimer leurs contributions sur les performances du réseau. L'application sur un réseau Pair-à-Pair réel dans une évaluation est compliquée et coûteuse à cause du nombre immense de

nœuds participants avec leurs arrivées et départs perpétuels dans le réseau. Pour cette raison que la plupart des travaux d'évaluation de performances se servent du principe de simulation [NLB⁺07]. Dans ce travail, Pastry/Scribe comprend le simulateur de réseau FreePastry, ce qui nous permet de simuler 1000 nœuds dans un seul ordinateur physique.

FreePastry ⁶est une implémentation open-source du protocole de Pastry en Java. La dernière version de FreePastry (publiée le 28 Janvier, 2003) comprend l'implémentation de PAST [DR01] un système d'archivage à base de Pastry et l'implémentation de Scribe. Il s'agit donc d'une application de Pastry, mais il peut aussi simuler un réseau Pastry. Il offre 3 choix de protocoles de transport pour l'application de l'utilisateur : direct, RMI, et Wire. Avec le protocole de transport direct, FreePastry simule un réseau avec un nombre des nœuds Pastry défini par l'utilisateur en une seule machine virtuelle Java. Avec le protocole RMI, Les réglages des paramètres de simulation, tels que le nombre de nœuds pour la simulation et le nombre d'événements à générer, se fait en fournissant les valeurs à la ligne de commande lors du démarrage des simulateurs locales. Freepastry garantit un équilibrage des charges du réseau puissant et une accessibilité aux documents rapide et facile.

3.3.2 Les métriques mesurées

La prise en considération de certaines métriques de performances aide à l'évaluation de notre approche de référencement dans le système Publier/Souscrire basé DHT. Parmi ces métriques, nous considérons le taux de churn, le taux de réussite de matching des publications aux souscriptions, le nombre de messages échangés dans les différents nœuds du réseau ainsi que le taux d'échec de matching.

3.3.2.1 Taux de churn

Le taux du churn décrit le nombre de connexions et de déconnexions par minute. Ainsi, pour obtenir un taux de churn fort il faut augmenter le nombre de changements dans le réseau pendant la période de simulation donc plus de changements nous

⁶<http://www.cs.rice.edu/CS/Systems/Pastry/FreePastry/>

avons plus le churn est élevé. Ainsi le processus de connexion et déconnexion se fait périodiquement et affecte tous les nœuds du réseau. De ce fait, suite à chaque changement dans la structure du réseau, une mise à jour s'effectue pour les tables de routage et le leafset. Pour bien évaluer notre approche nous avons testé les différents niveaux de taux de churn pour observer l'apport de l'approche dans le maintien de la disponibilité des tables de routage des souscriptions dans le service d'événement.

3.3.2.2 Taux de réussite de matching

Le taux de réussite de matching mesure le pourcentage de réussite de l'approche. Il est exprimé par le nombre de souscriptions correctement matchées avec la publication correspondante par rapport au nombre total des souscriptions. Cette métrique permet ainsi de nous indiquer en résultat le taux de la disponibilité des tables de routage des souscriptions.

$$\text{Taux de réussite} = \left(\frac{\text{Nombre-des-souscrits-matchés}}{\text{Nombre-total-des-souscrits}} \right) \times 100.$$

3.3.2.3 Trafic de messages

Cette métrique représente le nombre de messages échangés par les différents nœuds du réseau pour l'exécution de l'approche du référencement. Le résultat de cette métrique représente un indicateur de la réussite de matching entre les publications et les souscriptions et d'une légère surcharge due au processus de référencement. En effet, pour un nombre de références nul le matching ne se déroule pas vu l'augmentation du dynamisme du réseau et au fur et à mesure que nous ajoutons des références le taux d'acheminement des publications vers les souscriptions augmente donc c'est normal que le trafic des messages augmente mais concernant la surcharge du réseau due au référencement elle est pratiquement négligeable devant le grand apport de la réussite.

3.3.2.4 Taux d'erreur de matching

Le résultat de cette métrique est exprimé en pourcentage. Il est défini comme étant le nombre de souscrits ne recevant pas les publications désirées par rapport

au nombre de souscrits total. Cette métrique représente un indicateur du taux de la perte des tables de routage des souscriptions dans le réseau.

$$Taux - erreur = \left(\frac{Nombre-des-souscrits-non-matchés}{Nombre-total-des-souscrits} \right) \times 100.$$

3.3.3 Scénario

Nous avons implémenté notre approche de référencement dans le code source du système Publier/Souscrire P2P-TOPSS sous le simulateur intégré de Freepastry. Nous avons effectué des simulations afin d'évaluer et comparer les apports de notre stratégie sur les performances du réseau.

Les simulations sont effectuées dans un réseau disposant de 1000 nœuds dynamiques dans un espace d'identifiants de 128 bits. La durée de simulation du churn est égale à 340 secondes. A chaque période de 30 secondes, il y aura insertion d'une nouvelle souscription aléatoirement au niveau du réseau et à chaque période paramétrée selon le taux du churn, il y aura une déconnexion d'un nœud et une connexion d'un autre nœud afin que le réseau maintient toujours le même nombre de nœuds ainsi le nombre de connexion/déconnexion par période de simulation varie de 17 à 323 cela revient à une fréquence de connexion/déconnexion par minute de 3 à 57, et à chaque période de 60 secondes il y aura un multicast d'une publication.

Chaque simulation est répétée 10 fois et les valeurs indiquées représentent les moyennes.

3.3.4 Résultats et interprétations

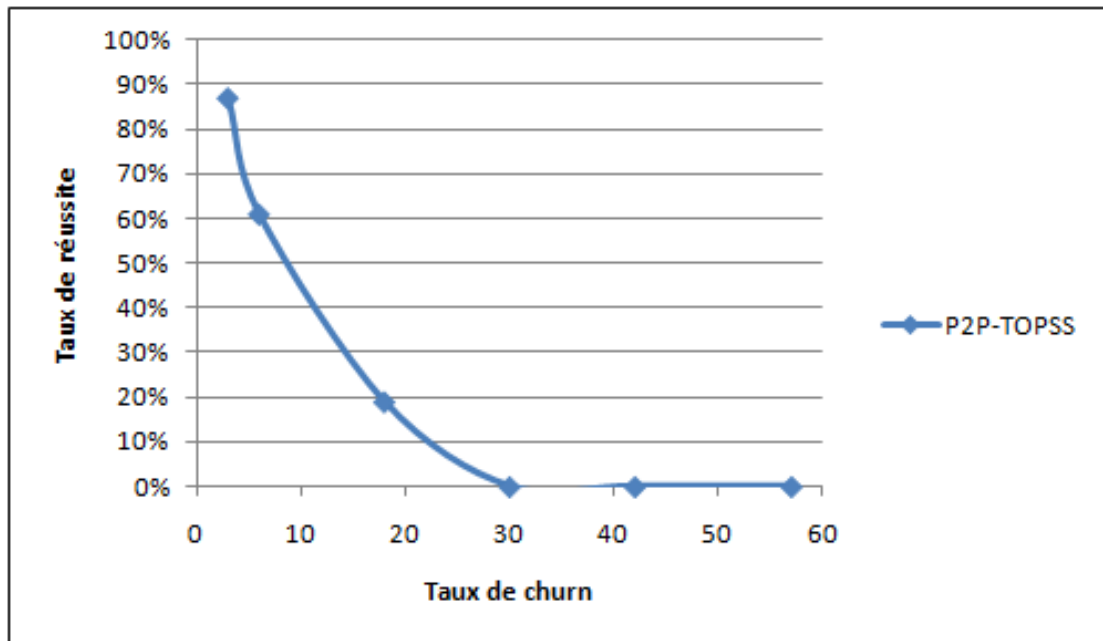


FIG. 3.6 – Dégradation de la disponibilité des souscriptions en fonction de l'augmentation du taux de churn

Le système P2P-TOPSS se caractérise par son support au dynamisme des nœuds. En fait ce support est limité par sa capacité de faire que les mises à jour des voisins lors d'un changement dans la structure des nœuds. En effet, ce système ne présente aucune résistance au niveau de la disponibilité des souscriptions lors des changements perpétuels des nœuds. Ainsi, la figure 3.6 illustre la dégradation de la disponibilité des souscriptions dans P2P-TOPSS avec l'augmentation du taux de dynamisme.

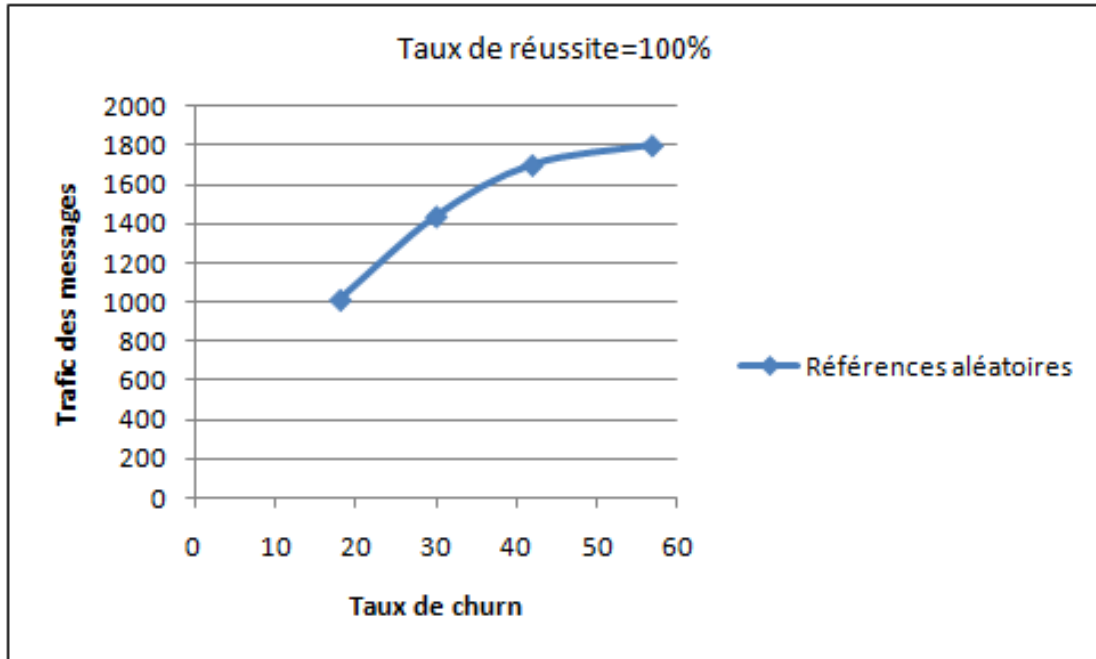


FIG. 3.7 – Référencement aléatoire

Cette limite présentée par P2P-TOPSS peut être résolue par la réplique des souscriptions. En fait, lors de l'acheminement d'une souscription vers la racine du sujet, autres chemins se lancent pour améliorer la disponibilité de cette souscription en la faisant acheminer plusieurs chemins. L'amélioration au niveau de la métrique de la réussite du matching présentée par la figure 3.7 n'est pas très efficace au niveau de la métrique du trafic des messages. En effet, l'application des références aléatoirement coûte très cher en termes de trafic de messages.

Nous proposons alors une autre idée inspirée de P-Grid qui permet d'améliorer le taux de réussite de matching tout en minimisant le trafic engendré par les références aléatoires.

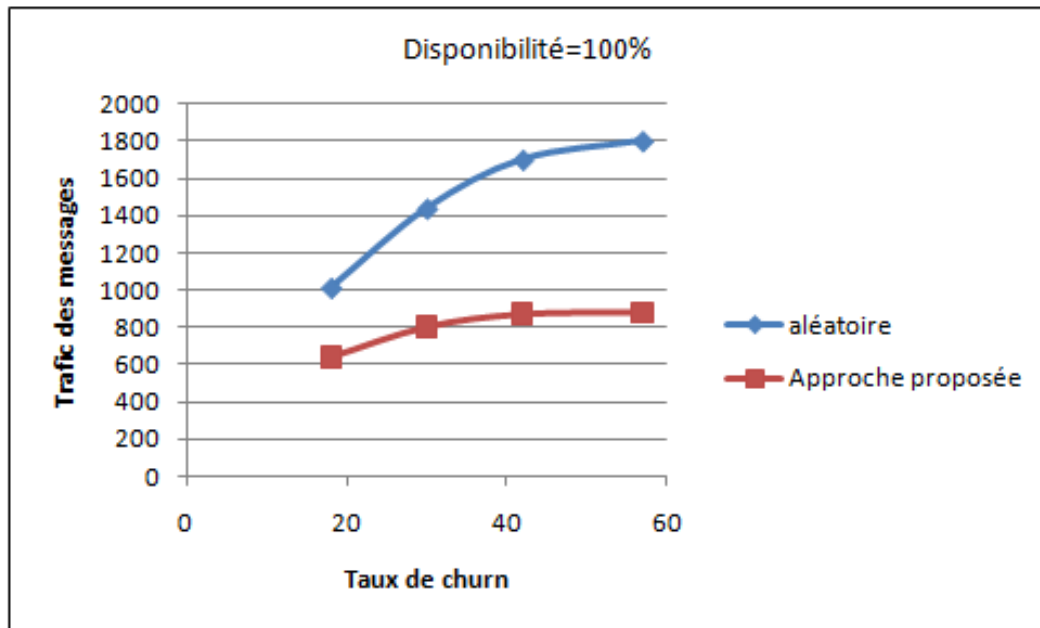


FIG. 3.8 – Comparaison du trafic moyen de messages en fonction du taux de churn

La figure 3.8 montre bien l'apport de la répartition des références selon le principe de P-Grid. Ainsi, un gain de l'ordre de 900 messages est obtenu pour un churn de 57/min. Ce qui montre bien que l'emplacement des références requiert bien un choix bien précis pour avoir le meilleur taux de réussite avec le moins de trafic surtout pour un dynamisme très fort.

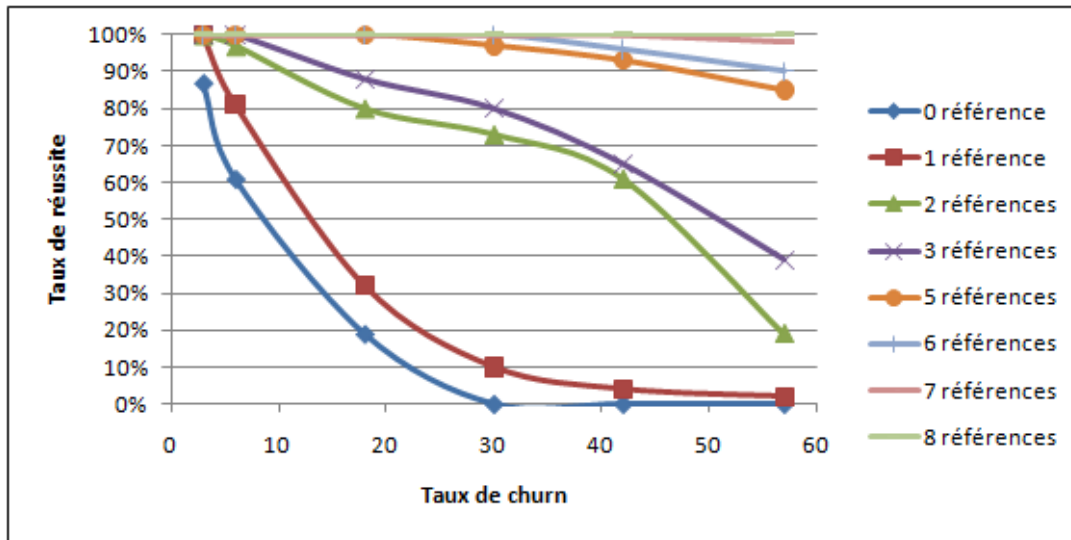


FIG. 3.9 – Comparaison du taux moyen de réussite en fonction du taux de churn

La figure 3.9 montre le taux moyen de réussite observé durant l'application de l'approche pour différents niveaux de churn. Le taux du churn décrit le nombre de connexions et de déconnexions par minute. Nous remarquons qu'à partir un taux de churn de 30/min le système P2P-TOPSS ne résiste plus et les tables de routage des souscriptions ne sont plus disponibles et que plus le taux de churn est élevé plus que ca demande de références pour atteindre le 100% de réussite. Nous constatons aussi que la disponibilité est beaucoup plus meilleure après l'application des références surtout à forte dynamicité. En fait, la dynamicité engendre de fréquentes pertes des tables de routage des souscriptions et P2P-TOPSS basé Scribe ne résiste pas à un taux élevé de dynamisme.

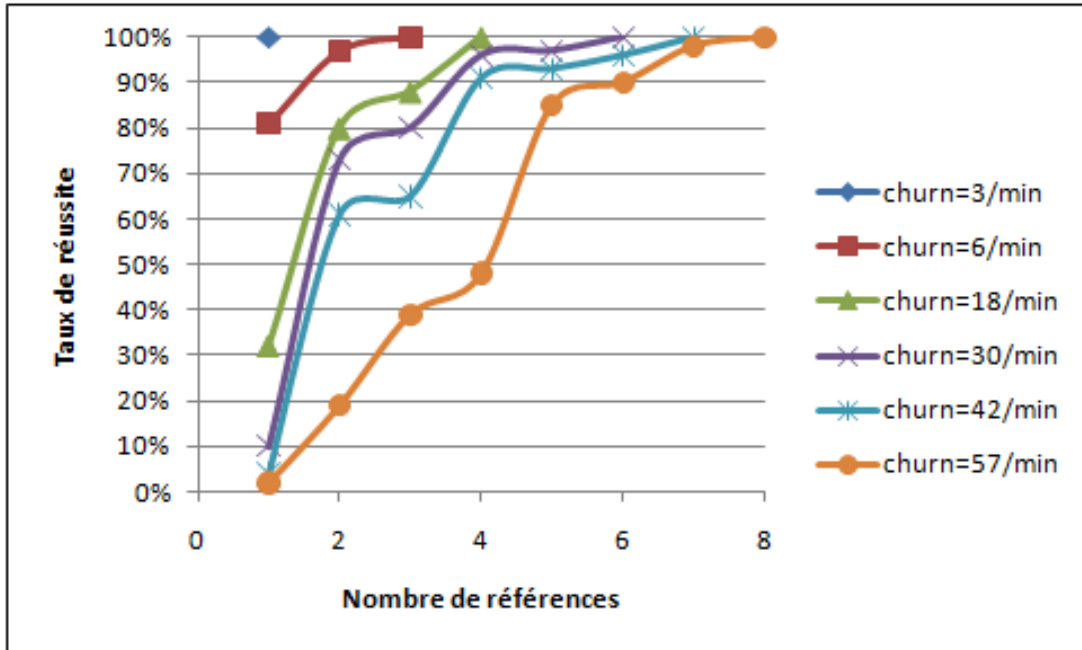


FIG. 3.10 – Taux moyen de réussite en fonction du nombre de références

La figure 3.10 montre le taux moyen de réussite de matching observé en fonction du nombre de références. Nous notons qu'un taux de réussite égal à 100% est atteint lorsque chaque souscription se réfère à 6 références comme le montre la figure 3.10 pour un taux de churn = 30/minute, et ce nombre n'est pas gênant en terme de trafic moyen de messages.

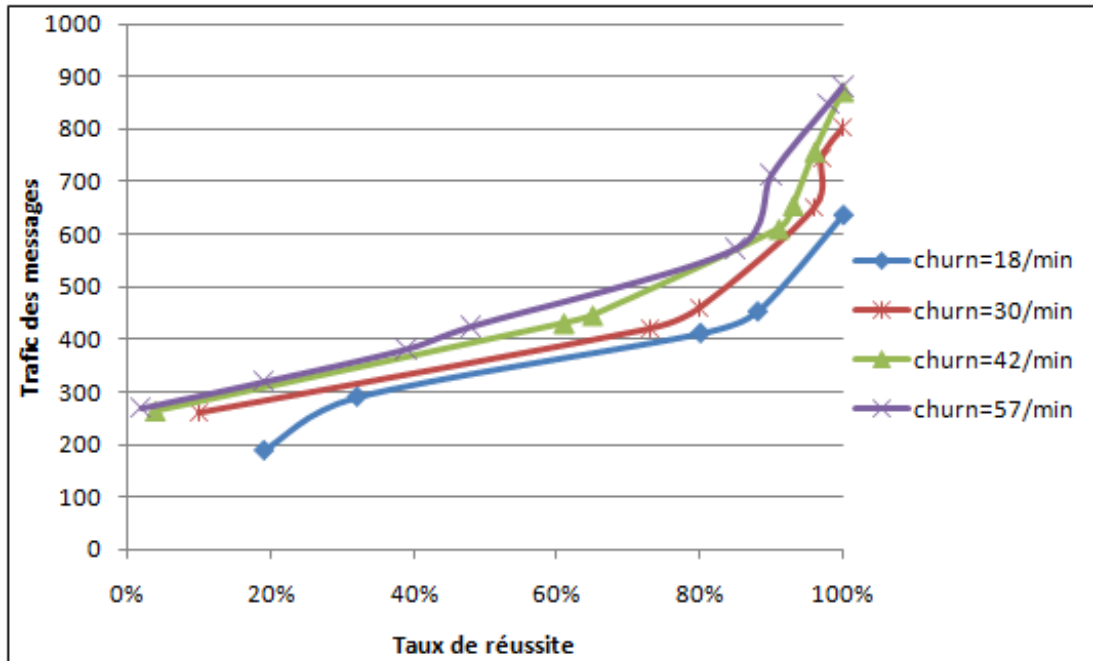


FIG. 3.11 – Trafic moyen de messages en fonction du taux de réussite

La figure 3.11 représente le trafic moyen de messages produit par notre stratégie de référencement en fonction du taux de réussite. Nous remarquons que le trafic de messages présente une augmentation proportionnelle avec l'élévation du taux de réussite souhaité pour chaque niveau de churn. Pour atteindre un taux de réussite égal à 100%, il fallait un trafic moyen de transfert de messages de 800 messages comme le montre la figure 3.11 pour un taux de churn égal à 30/minute.

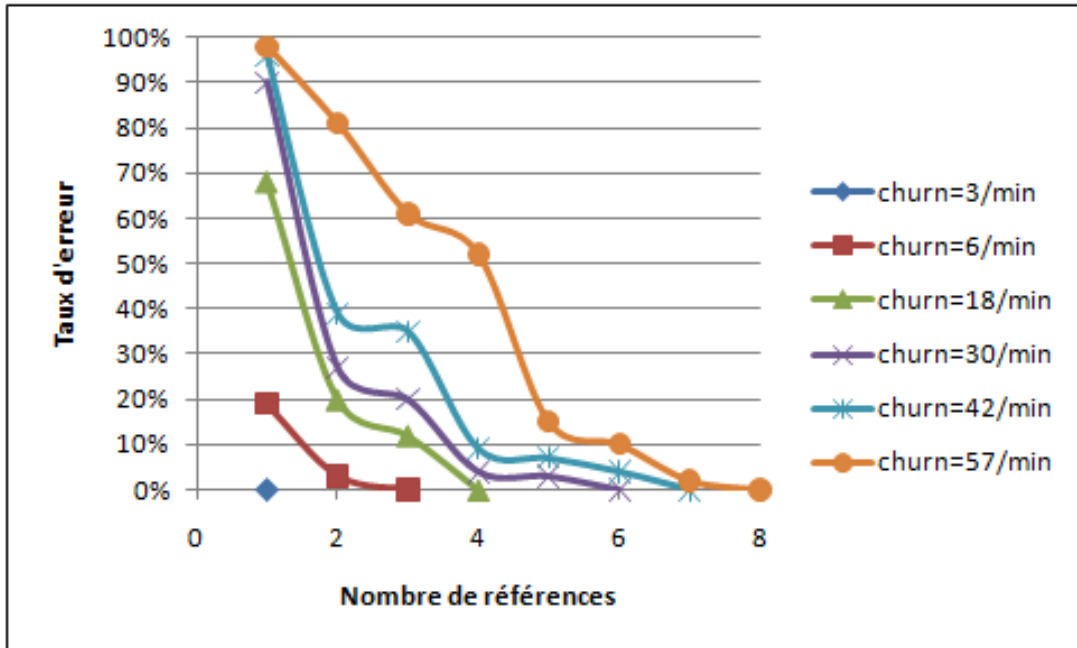


FIG. 3.12 – Taux moyen d'erreur en fonction du nombre de références

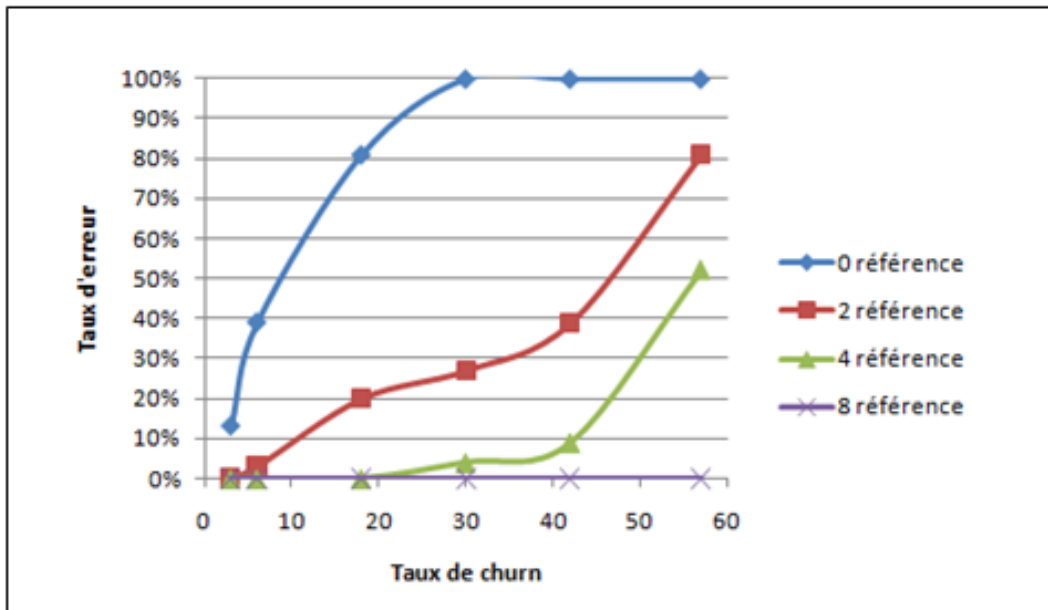


FIG. 3.13 – Comparaison du taux d'erreur observé en fonction du taux de churn

La figure 3.13 résume le taux moyen d'erreur observé en fonction de différents niveaux de dynamisme considérés. Nous constatons que le taux d'erreur propre à notre approche est inférieur à celui observé avant l'application du référencement, c'est-à-dire le nombre de souscriptions non matchés par les publications en appliquant la stratégie de référencement est plus élevé que celui obtenu sans l'application des références. Ce résultat est expliqué, comme déjà montré dans l'interprétation de la figure 3.9, par le fait que les références augmentent la disponibilité des données dans un environnement caractérisé par le dynamisme.

3.4 Conclusion

Dans ce chapitre, nous avons introduit notre nouvelle approche de résistance au churn qui a été intégrée dans le système Scribe dans le but d'améliorer ses performances. Ainsi, avons montré que la contribution de notre approche se présente surtout dans son utilisation dans un réseau très dynamique. Ainsi, nous avons présenté les résultats de simulation de notre approche de référencement implémentée sur le système Publier/Souscrire P2P-TOPSS et nous avons prouvé les corrections fournies par notre approche sur les différentes métriques suggérées.

CONCLUSION GÉNÉRALE

Les réseaux DHT offrent une solution efficace face au problème de localisation de données contribuant au passage à l'échelle. Dans ce contexte, le développement de plusieurs protocoles a été élaboré notamment Chord, Pastry, Tapestry, CAN. Toutefois, ces systèmes sont caractérisés par un dynamisme très fort ce qui provoque des perturbations dans le réseau. Ce problème de dynamisme a été évoqué surtout au niveau du routage mais il n'a pas eu beaucoup d'intérêt au niveau de stockage des données. En fait, quand un nœud quitte le système, toutes ses données par la suite sont perdues. D'où la nécessité d'avoir des politiques permettant d'assurer la disponibilité des données à n'importe quel moment dans le réseau DHT dynamique.

Dans ce projet, nous nous sommes intéressées à la résistance au churn dans les systèmes Publier/Souscrire basés DHT. Pour réaliser une telle résistance, nous avons suggéré une approche ayant le référencement des souscriptions comme principe fondamental. L'exécution du référencement se fait à chaque élaboration d'une souscription offrant une meilleure disponibilité des données pour un réseau subissant des dynamismes au niveau de ses nœuds.

Notre approche améliore la probabilité d'acheminement des publications vers les souscriptions malgré le changement perpétuel de la structure du réseau. Pour l'évaluation de notre travail nous avons considéré le middleware P2P-TOPSS. Nous avons déployé ce middleware dans le but d'améliorer sa résistance face au dynamisme du réseau.

Bien que la solution proposée améliore de façon considérable les performances des systèmes Publier/Souscrire déployés sur un réseau caractérisé par un dynamisme perpétuel, l'étude profonde de différents aspects doit être élaborée. Comme perspectives, nous visons à optimiser l'algorithme de référencement proposé pour diminuer le trafic ajouté. Nous proposons aussi une phase de monitoring pour la localisation des dispatchers les plus dynamiques et les données les plus pertinentes pour y appliquer l'algorithme de référencement. Par la suite, nous pouvons ajouter d'autres

métriques de performances pour notre système publier/souscrire. Nous proposons, aussi, de gérer la cohérence des souscriptions répliquées c'est-à-dire si un souscrit met à jour sa souscription, alors cette mise-à-jour doit être propagée aux autres références.

Bibliographie

- [ACMD⁺03] Karl Aberer, Philippe Cudré-Mauroux, Anwitaman Datta, Zoran Despotovic, Manfred Hauswirth, Magdalena Puceva, and Roman Schmidt, *P-grid : a self-organizing structured p2p system.*, SIGMOD Record (2003), 29–33.
- [AH00] Eytan Adar and Bernardo A. Huberman, *Free riding on gnutella.*, First Monday (2000), –1–1.
- [BPS05] Jean-Michel Busca, Fabio Picconi, and Pierre Sens, *Pastis : A highly-scalable multi-user peer-to-peer file system.*, Euro-Par’05, 2005, pp. 1173–1182.
- [CDKR02] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron, *Scribe : A large-scale and decentralized application-level multicast infrastructure*, IEEE Journal on Selected Areas in Communications (JSAC **20** (2002)), 2002.
- [CF05] Paolo Costa and Davide Frey, *Publish-subscribe tree maintenance over a dht.*, ICDCS Workshops’05, 2005, pp. 414–420.
- [CMM02] Russ Cox, Athicha Muthitacharoen, and Robert Morris, *Serving dns using a peer-to-peer lookup service.*, IPTPS’02, 2002, pp. 155–165.
- [CNF98] Gianpaolo Cugola, Elisabetta Di Nitto, and Alfonso Fuggetta, *Exploiting an event-based infrastructure to develop complex distributed systems.*, ICSE’98, 1998, pp. 261–270.
- [CRW01] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf, *Design and evaluation of a wide-area event notification service.*, ACM Trans. Comput. Syst. (2001), 332–383.
- [DGMP08] Olivier Dalle, Frédéric Giroire, Julian Monteiro, and Stéphane Pérennes, *Analysis of Failure Correlation in Peer-to-Peer Storage Systems*, Research Report RR-6771, INRIA, 2008.
- [DKK⁺01] Frank Dabek, M. Frans Kaashoek, David R. Karger, Robert Morris, and Ion Stoica, *Wide-area cooperative storage with cfs.*, SOSP’01, 2001, pp. 202–215.
- [DLS⁺04] Frank Dabek, Jinyang Li, Emil Sit, James Robertson, M. Frans Kaashoek, and Robert Morris, *Designing a dht for low latency and high throughput.*, NSDI’04, 2004, pp. 85–98.
- [DR01] Peter Druschel and Antony I. T. Rowstron, *Past : A large-scale, persistent peer-to-peer storage utility.*, HotOS’01, 2001, pp. 75–80.

- [FM08] Davide Frey and Amy L. Murphy, *Failure-tolerant overlay trees for large-scale dynamic networks.*, Peer-to-Peer Computing'08, 2008, pp. 351–361.
- [GGG⁺03] P. Krishna Gummadi, Ramakrishna Gummadi, Steven D. Gribble, Sylvia Ratnasamy, Scott Shenker, and Ion Stoica, *The impact of dht routing geometry on resilience and proximity.*, SIGCOMM'03, 2003, pp. 381–394.
- [HAY⁺05] Ragib Hasan, Zahid Anwar, William Yurcik, Larry Brumbaugh, and Roy H. Campbell, *A survey of peer-to-peer storage techniques for distributed file systems.*, ITCC (2)'05, 2005, pp. 205–213.
- [ID01] T. Rowstron Antony I. and Peter Druschel, *Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility.*, SOSP'01, 2001, pp. 188–201.
- [KBC⁺00] John Kubiawicz, David Bindel, Yan Chen, Steven E. Czerwinski, Patrick R. Eaton, Dennis Geels, Ramakrishna Gummadi, Sean C. Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Y. Zhao, *Oceanstore : An architecture for global-scale persistent storage.*, ASPLOS'00, 2000, pp. 190–201.
- [LKR03] Dmitri Loguinov, Anuj Kumar, Vivek Rai, and Sai Ganesh, *Graph-theoretic analysis of structured peer-to-peer systems : routing distances and fault resilience.*, SIGCOMM'03, 2003, pp. 395–406.
- [LMSM09] Sergey Legtchenko, Sébastien Monnet, Pierre Sens, and Gilles Muller, *Churn-resilient replication strategy for peer-to-peer distributed hash tables.*, SSS'09, 2009, pp. 485–499.
- [LNBK02] David Liben-Nowell, Hari Balakrishnan, and David R. Karger, *Analysis of the evolution of peer-to-peer systems.*, PODC'02, 2002, pp. 233–242.
- [LSG⁺04] Jinyang Li, Jeremy Stribling, Thomer M. Gil, Robert Morris, and M. Frans Kaashoek, *Comparing the performance of distributed hash tables under churn.*, IPTPS'04, 2004, pp. 87–99.
- [MM02] Petar Maymounkov and David Mazières, *Kademlia : A peer-to-peer information system based on the xor metric.*, IPTPS'02, 2002, pp. 53–65.
- [MMGC02] Athicha Muthitacharoen, Robert Morris, Thomer M. Gil, and Benjie Chen, *Ivy : A read/write peer-to-peer file system.*, OSDI'02, 2002, pp. –1–1.
- [NLB⁺07] S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, Ian Wakeman, and Dan Chalmers, *The state of peer-to-peer simulators and simulations.*, Computer Communication Review (2007), 95–98.
- [OPSS93] Brian M. Oki, Manfred Pflügl, Alex Siegel, and Dale Skeen, *The information bus - an architecture for extensible distributed systems.*, SOSP'93, 1993, pp. 58–68.
- [PRR97] C. Greg Plaxton, Rajmohan Rajaraman, and Andréa W. Richa, *Accessing nearby copies of replicated objects in a distributed environment.*, SPAA'97, 1997, pp. 311–320.

- [RD01] Antony I. T. Rowstron and Peter Druschel, *Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems.*, Middleware'01, 2001, pp. 329–350.
- [RFH⁺01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard M. Karp, and Scott Shenker, *A scalable content-addressable network.*, SIGCOMM'01, 2001, pp. 161–172.
- [RGK⁺05] Sean C. Rhea, Brighten Godfrey, Brad Karp, John Kubiawicz, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Harlan Yu, *Opendht : a public dht service and its uses.*, SIGCOMM'05, 2005, pp. 73–84.
- [SMLN⁺03] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan, *Chord : a scalable peer-to-peer lookup protocol for internet applications.*, IEEE/ACM Trans. Netw. (2003), 17–32.
- [TAJ03] David K. Tam, Reza Azimi, and Hans-Arno Jacobsen, *Building content-based publish/subscribe systems with distributed hash tables.*, DBISP2P'03, 2003, pp. 138–152.
- [YGM01] Beverly Yang and Hector Garcia-Molina, *Comparing hybrid peer-to-peer systems.*, VLDB'01, 2001, pp. 561–570.
- [ZHS⁺04] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiawicz, *Tapestry : a resilient global-scale overlay for service deployment.*, IEEE Journal on Selected Areas in Communications (2004), 41–53.
- [ZZJ⁺01] Shelley Zhuang, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, and John Kubiawicz, *Bayeux : an architecture for scalable and fault-tolerant wide-area data dissemination.*, NOSSDAV'01, 2001, pp. 11–20.



Traitement de la Forte Dynamicit  dans un Syst me Publier/Souscrire Bas  DHT

Fatma ABDENNADHER

الخلاصة: ديناميكية الشبكات يمثل تحديا لمصممي أنظمة النشر والترسيم. و بالتالي، فإن هذه التغييرات المستمرة في هيكل الشبكة المزدوجة يسبب فقدان البيانات و عدم الاستجابة إلى مطالب المشتركين. وفي هذا المنظور، يندرج مشروعنا الذي نقترح فيه منهجاً جديداً يعتمد على نظام المرجعية لمقاومة الديناميكية العالية في أنظمة النشر والترسيم المستندة على الموضوع والمنظمة حسب جدول التشفير الموزع.

R sum  : La dynamique des r seaux pr sente un d fi pour les concepteurs des syst mes Publier/Souscrire. Ainsi, de telles modifications perp tuelles dans la structure du r seau pair- -pair causent la perte des messages et la non satisfaction des souscrits. C'est dans cette perspective que se situe notre projet de mast re, qui consiste   proposer une approche de r f rencement pour la r sistance au churn (forte dynamique), appliqu e sur un syst me Publier/Souscrire bas  sujet et structur  au dessus d'une table de hachage distribu e (DHT).

Abstract: Dynamicity of networks presents a challenge for the designers of Publish / Subscribe systems. Thus, these constant changes in the structure of the P2P network cause the loss of data and the non-satisfaction of subscribers. In this perspective, that our master project is, in which we propose an approach of references to resist face to the churn (high dynamicity), applied on a topic-based Publish/Subscribe system structured over a Distributed Hash Table (DHT).

المفاتيح: النشر و الترسيم ، المزدوجة، جدول التشفير الموزع ، الديناميكية العالية، نظام المرجعية.

Mots cl s: Publier/Souscrire, pair- -pair, DHT, forte dynamique, r f rencement.

Key-words: Publish/Subscribe, P2P, DHT, churn, references.