Méthode décentralisée de mesure de dynamisme d'un réseau logique

Vincent Gramoli, Anne-Marie Kermarrec, Erwan Le Merrer

INRIA Futurs Saclay, 2-4 rue Jacques Monod 91893 Orsay Cedex, France
IRISA, Campus Universitaire de Beaulieu, 35042 Rennes Cedex, France
vincent.gramoli@inria.fr, {akermarr,elemerre}@irisa.fr

Résumé

L'une des contraintes majeures qui caractérise les réseaux à grande échelle est leur dynamisme. À l'instar des systèmes pair-à-pair, ces réseaux subissent un flux permanent d'arrivées et de départs de leurs participants. Ce dynamisme est dû au grand nombre de nœuds participants qui sont indépendamment assujettis à des pannes, des connexions ou déconnexions. De nombreuses applications distribuées font l'hypothèse d'un taux maximal de départs et d'arrivées. Pour tolérer le dynamisme, ces applications proposent un mécanisme dont la complexité dépend de cette borne maximale et ne prend pas en compte les variations possibles de ce taux. Il apparaît donc nécessaire *(i)* d'évaluer précisément ce taux qui caractérise l'intensité du dynamisme, aussi appelé *va-et-vient* (de l'anglais *churn*) et *(ii)* d'évaluer ces variations au cours du temps. Nous présentons, à notre connaissance, la première méthode générique d'estimation distribuée du va-et-vient. Cette méthode est dite générique, car elle est applicable quelle que soit la topologie logique du réseau sous-jacent. Notre technique tolère le passage à grande échelle puisque qu'elle ne s'appuie que sur l'étude du voisinage direct de chaque nœud, avec un emploi d'informations d'une taille constante sur chaque nœud. Des simulations préliminaires montrent la précision des estimations produites, malgré le départ massif des nœuds.

Mots-clés : Réseaux à grande échelle, dynamisme, monitoring, algorithmes épidémiques

1. Introduction

1.1. Contexte et motivations

De nos jours, les systèmes à grande échelle prennent une place de plus en plus importante dans la recherche en systèmes distribués. Les systèmes à grande échelle sont des systèmes distribués possédant un très grand nombre de participants ; le système de communication sur IP Skype, comptant actuellement 10 millions de personnes connectées simultanément au réseau logique, en est un exemple. Les participants (ou *nœuds*) de tels systèmes sont sujets à des pannes, des connexions ou déconnexions inopinées, souvent dues à des facteurs environnementaux incontrôlables et imprévisibles. La taille de ces systèmes les rend donc intrinsèquement dynamiques, puisqu'il est très fréquent de voir plusieurs nœuds se connecter et se déconnecter quasi-simultanément.

Afin de pallier ce dynamisme, les solutions proposées émettent l'hypothèse d'un maximum de départs et d'arrivées par période de temps fixée. Cette borne supérieure caractérise une intensité de dynamisme aussi appelée *va-et-vient* [10]. Avec cette hypothèse, le système est capable de se réadapter afin d'assurer les propriétés désirées. Par exemple, la table de hachage distribuée (DHT) Pastry [21] suppose que moins de $k/2$ nœuds, adjacents dans l'espace d'identification, quittent quasi-simultanément le système (avec k se situant typiquement entre 16 et 32). De plus, Kuhn et al. [13] supposent qu'au plus $O(\log N)$ pairs quittent le système par unité de temps, ou N est la taille du système à cet instant. Ainsi le va-et-vient toléré est choisi de façon arbitraire, car on ignore sa valeur réelle dans l'environnement considéré.

Un article [9] a récemment montré l'impact que pouvait avoir la connaissance du va-et-vient sur les performances du système. Plus précisément, les auteurs ont montré la probabilité qu'une donnée persiste, ainsi que la façon d'adapter la duplication en fonction de différentes valeurs de va-et-vient. Afin de savoir précisément quel est le va-et-vient qu'un système doit pouvoir tolérer, de nombreux travaux ont tenté de caractériser de manière centralisée la distribution des durées de vie des nœuds sur certains

systèmes particuliers ; plus récemment des solutions distribuées de mesure ont été proposées pour un type particulier de topologie réseau (DHT).

1.2. État de l'art

De nombreux articles se proposent de caractériser la disponibilité des participants des systèmes pair-à-pair ; citons par exemple [2], [22], et [23], qui traitent des réseaux Overnet pour le premier ; Kad, Gnutella et BitTorrent pour le second ; et enfin Maze pour le dernier. Ces études sont menées de manière centralisée, par lecture de fichiers de *logs* au réseau de pairs ou par exploration récursive de l'espace de localisation des pairs, puis par envoi de requêtes régulières (*pings*) pour analyser leur temps de présence sur le réseau logique.

Ces différents travaux permettent à des développeurs d'applications distribuées de calibrer leur tolérance au dynamisme. Les limitations de ces approches sont *(i)* de présenter des résultats figés dans le temps, et souvent représentatifs de tendances générales, et *(ii)* de présenter des observations concernant uniquement des applications de distribution de contenu, ou de partage de fichiers, alors qu'il est assez naturel de penser que les caractéristiques d'emploi d'un réseau par l'utilisateur varient en fonction du type d'application portée (*e.g.* téléphonie, calcul).

Ces études servent ainsi à paramétrer statiquement les applications déployées, en fonction de tendances observées, mais ne permettent pas d'obtenir des informations en temps réel sur le va-et-vient, exploitables par d'autres applications distribuées.

Certaines approches se proposent naturellement de tirer parti de la connaissance de la topologie logique sous-jacente. Les méthodes [7, 3] se basent sur les caractéristiques des réseaux structurés, pour mesurer les temps de présence en/hors ligne des pairs. L'article [3], paru en 2007, nécessite le retour systématique des pairs après leur déconnexion du réseau, et propose une appoximation des temps de sessions des nœuds observés localement. Ces articles s'appuient sur le fait que les évènements observés dans une sous partie d'un réseau structuré sont représentatifs du comportement global du réseau (par extrapolation). Cette dernière hypothèse rend malheureusement ces méthodes non applicables pour des réseaux à structure arbitraire.

1.3. Contributions

Ce papier propose, à notre connaissance, la première évaluation distribuée générique du va-et-vient d'un système dynamique. L'algorithme présenté est exécuté par les participants du système de telle manière que chacun obtient sa propre estimation du va-et-vient global du système. Par conséquent, cette information peut être directement réutilisée par une application distribuée afin de s'adapter pour tolérer le va-et-vient du système couramment observé. Cet algorithme ne fait l'hypothèse d'aucune structuration particulière du réseau de communication. Les participants communiquent en utilisant un mécanisme épidémique [4] permettant une diffusion très rapide de l'information au sein du réseau [12].

La solution proposée dans ce papier consiste pour un nœud, à comptabiliser les départs et arrivées parmi les nœuds auxquels il est connecté, aussi appelés ses *voisins*. Ces observations locales sont collectées par chaque nœud durant une période de temps initialement fixée et sont ré-effectuées à intervalles de temps réguliers. Une fois ces observations effectuées, elles sont agrégées sur l'ensemble du système par un mécanisme de transmission épidémique. Cette agrégation est effectuée entre l'information que le nœud possède et l'information qu'il reçoit, afin que la taille de l'information stockée localement et celle de l'information envoyée dans un message soient négligeables par rapport à la taille du système.

2. Modèle considéré

Cette section présente les hypothèses et notations employées pour la description de la solution.

2.1. Réseau logique de communication

Le réseau logique de machines participant à l'application distribuée est caractérisable par un graphe de communication connecté $\mathcal{G}$, composé de N nœuds et de E arêtes. Chaque nœud i communique de façon synchrone avec son voisinage direct. Les canaux de communication sont supposé fiables, ainsi nous supposons qu'aucun message n'est perdu, dupliqué ou créé. La relation de voisinage est symétrique (si i est un voisin de j alors j est également un voisin de i), résultant ainsi en un graphe non orienté. Le nombre de voisins d'un nœud i est appelé son *degré*, et est noté d_i. Afin de limiter le degré de chaque

nœud, un nœud i peut refuser une relation de voisinage avec un autre nœud j demandeur.

2.2. Va-et-vient ou intensité du dynamisme

Chaque nœud peut quitter le système à tout instant et un nœud hors du système peut joindre celui-ci à tout instant. L'intensité du dynamisme, appelée *va-et-vient*, représente le ratio d'arrivées et de départs de nœuds sur la taille de la population initiale, par unité de temps. Le va-et-vient est donc compris entre 0 et 1. Dans notre modèle, un départ représente indifféremment un départ volontaire ou involontaire (*e.g. panne*) ; un nœud qui quitte le système n'a pas à en aviser d'autres nœuds.
La détection d'une arrivée sur le réseau logique est rendue possible dans la mesure où le nœud entrant initie une procédure d'accès en contactant des nœuds déjà présents afin de créer de nouvelles relations de voisinage.
Notons que notre méthode comptabilise les nœuds n'ayant pas quitté le réseau logique, mais étant surchargés et ne pouvant par conséquent plus participer au protocole, faute de ressources.

2.3. Maintenance du réseau

Nous ne traitons pas de la manière dont est maintenu le réseau logique par un éventuel protocole sous-jacent, pour le conserver connecté ou conserver les propriétés de la structure logique par exemple. Ce maintien de la connectivité peut être assuré en utilisant un des algorithmes existant [19, 20, 14]. Dans la suite de cet article, nous nous intéressons au calcul du va-et-vient, sans influer sur la connectivité du réseau logique.

2.4. Asynchronie et détecteurs de défaillance

Le modèle proposé ci-dessus suppose une communication fiable et synchrone. Ainsi, nous proposons une implémentation simple d'un *détecteur de défaillance parfait*. Celui-ci utilise un mécanisme de *battement de cœur* (de l'anglais *heartbeat*). Chaque nœud i du système envoie un message a chacun de ses voisins j de façon périodique. En l'absence de réception d'un tel message, j détecte que i a quitté le système (i.e., i est tombé en panne ou s'est volontairement déconnecté du système). En système asynchrone, le délai de transmission d'un message peut être arbitrairement long, empêchant la détection des défaillances au moyen de ce mécanisme. Il existe des détecteurs de défaillance plus faibles qu'un détecteur parfait et susceptibles de rendre l'algorithme correct en système asynchrone. La recherche du plus faible détecteur de défaillance ne fait pas l'objet de cet article.

2.5. Prérequis

Dans des réseaux logiques où un canal de communication est employé de manière persistante entre deux voisins, il n'existe pas (du point de vue local aux nœuds) de différence entre le départ d'un participant du réseau et une simple réorganisation de voisinage, car nous nous plaçons dans le pire scénario où les nœuds quittent le réseau logique sans préavis. Certains nœuds réorganisent leur voisinage direct suivant des heuristiques, comme par exemple pour converger vers un maillage logique reflétant des caractéristiques de latences physiques inter-nœuds [15]. Une information supplémentaire est alors nécessaire dans notre heuristique afin de différencier ces deux cas de figure ; le nœud effectuant une réorganisation d'un de ses liens de voisinage envoie donc à son contact un message (nous l'appellerons *message de réorganisation*) lui annonçant qu'il ne s'agit pas d'un départ du réseau logique. Une rupture de lien pour laquelle aucun message n'a été reçu est ainsi interprétée comme un départ par le nœud détectant la disparition au sein de sa liste de voisins. Notre mécanisme s'appuie sur le fait que chaque nœud puisse déterminer à un instant T son propre degré, et puisse le communiquer à ses voisins directs en cas de variation.

3. Objectifs et notations associées

Nous présentons maintenant les notations nécessaires à la définition de notre problématique. Comme mentionné précédemment, la taille du graphe de communication $\mathcal{G}$ est noté N. Notons N_T la taille du système à l'instant T. Le cœur du système, noté σ, représente l'ensemble des nœuds présents au temps T durant une période δ. L'ensemble des nœuds qui sont arrivés dans le système entre T et $T+\delta$ est noté a. De façon similaire, l'ensemble des nœuds ayant quitté le système entre les temps T et $T+\delta$ est dénoté q. Ainsi, nous obtenons $N_{T+\delta} = N_T - card(q) + card(a)$, avec $N_T = card(\sigma) + card(q)$. La figure 1

présente l'évolution du système, avec ces notations, aux temps T et $T + \delta$.

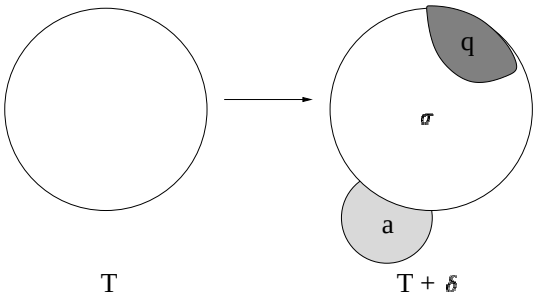

FIG. 1 – Définition des ensembles de nœuds résultant d'une évolution du réseau entre le temps T et $T + \delta$; l'ensemble a représente les nœuds arrivés à $N_{T+\delta}$, q ceux disparus, et σ le noyau (nœuds ni arrivés ni disparus)

L'objectif de cet article est d'estimer $\frac{card(a)}{N_T}$ et $\frac{card(q)}{N_T}$, valeurs qui représentent respectivement la proportion de nœuds arrivés dans le système et celle ayant quitté le système, pendant une période δ, entre les temps T et $T+\delta$. Par exemple, si cette estimation de départ est de 0.5, cela signifie que 50% des nœuds présents au temps T ont quitté le système au temps $T + \delta$. Remarquons que si aucune arrivée a eu lieu durant cette période, alors $N_{T+\delta} = N_T * 0.5$.
Nous employons ainsi deux indicateurs, l'un concernant les arrivées et l'autre les départs. Nous ne décrivons ainsi pas le dynamisme du système avec un seul indicateur simple, qui aurait pu représenter la variation de taille entre T et $T + \delta$ (*e.g.* +0.5 ou -0.1), car ce delta $\frac{|card(a)-card(q)|}{N_T}$ après un temps δ gomme le dynamisme réel du réseau. Imaginons par exemple le cas $card(a) = card(q)$ (*i.e.* le même nombre de nœuds sont entrés et sortis du système), reflétant une variation de taille de N égale à 0, alors que celle ci a un impact très probable sur la perte de répliquas d'un réseau, sur le besoin d'accroître ce degré de duplication pour supporter la charge des nouveaux entrants, ou sur la réorganisation de la structure du réseau logique.

4. Algorithme d'évaluation du va-et-vient du système

Nous décrivons maintenant notre méthode d'obtention de ces informations de va-et-vient. Le procédé s'exécute au sein de chaque nœud du système à analyser, et démarre périodiquement, ou après une notification particulière (*e.g.* ordre de l'administrateur du réseau par diffusion globale). L'algorithme est composé de deux phases ; la première est une phase de mesure du dynamisme local à chaque nœud pendant une durée δ, alors que la seconde est une phase de réduction de variance des estimations locales, afin d'obtenir une estimation du dynamisme correcte au niveau global du réseau.

4.1. Phase de mesure du va-et-vient local

Pour les besoins de la mesure locale, chaque nœud héberge et gère deux variables, q_{local} et a_{local}, représentant respectivement la proportion de nœuds de son voisinage direct ayant quitté le réseau logique, et la proportion de nœuds entrant dans le système par contact immédiat. Ces deux compteurs sont initialisés à 0 au début de la phase de mesure.
Selon notre protocole, un nœud i avise ses voisins directs de son degré lors de son arrivée sur le réseau. Si ce degré est modifié (un nœud est arrivé ou parti du voisinage de i), i propage l'information à jour à l'ensemble de son voisinage. Nous obtenons ainsi un réseau dans lequel chaque participant connaît à chaque instant le degré de tout ses voisins. La *vue* d'un nœud i (notée V_i) peut alors être définie comme un ensemble de tuples $\langle j, d_j \rangle$ (où j est un voisin direct du nœud), pour l'ensemble des voisins de i.
La mesure débute au temps T et se termine à $T + \delta$; dans le cas d'un processus de mesure continu, le procédé est redémarré toutes les δ unités de temps.

4.1.1. Départ d'un nœud

Un départ est détecté localement par un nœud voisin s'il ne reçoit pas de message indiquant une réorganisation de lien, et pas de message de battement de cœur au delà d'un seuil de temps prédéfini Δk,

où Δ est la période d'envoi de battement. Ce paramètre k permet de supporter des pertes de messages pour les protocoles d'envois de messages non fiables, de tenir compte d'une éventuelle surcharge de nœuds ou d'une indisponibilité temporaire du médium de communication entre les deux contacts. Un détecteur de défaillance alternatif pourra bien entendu être préféré ici par l'utilisateur.
L'incrément du compteur q_{local} sur un nœud i dépend alors du degré du nœud j considéré comme disparu : $q_{local\_i} \leftarrow q_{local\_i} + \frac{1}{d_j}$. Comme chaque voisin du nœud j détecte son départ à terme, un tel incrément sur chaque voisin permet de conserver une information correcte au sein du réseau, et ce par addition ($\Sigma_1^{d_j} \frac{1}{d_j} = 1$) ; l'information globale du départ d'un nœud est alors conservée sur le réseau. Ainsi, si un nœud de degré $d = 1$ quitte le réseau, son unique voisin incrémente alors son compteur d'une unité ($+\frac{1}{1}$), reflétant bien l'information exacte ; s'il avait possédé un $d = 2$, alors deux nœuds auraient procédé à un incrément d'un demi.
Notons qu'un éventuel message d'un nœud annonçant son désir de quitter le réseau logique irait dans le sens de l'amélioration des performances de la méthode, sur le plan de la latence de détection, comme sur celui de l'économie de ressources qui auraient été nécessaires pour procéder à une détection d'un départ sans préavis.

4.1.2. Arrivée d'un nœud

La procédure de surveillance des arrivées fonctionne de manière similaire : un nœud contacté pour une relation de voisinage, s'il ne reçoit pas de son contact de message de réorganisation, considère cet évènement comme une nouvelle entrée dans le système. Deux options sont alors possibles pour l'incrément de a_{local} ; le nœud entrant i peut choisir son degré d_i désiré en fin de son processus d'accès au réseau, et ainsi l'envoyer à ses nouveaux voisins, ceux-ci incrémentant leur a_{local} de la sorte : $a_{local} \leftarrow a_{local} + \frac{1}{d_i}$. La seconde option, dans le cas où l'entrant ne peut prévoir son degré final, est de faire incrémenter le premier nœud avec qui il entre en contact pour voisinage de $a_{local} \leftarrow a_{local} + 1$, puis d'envoyer des messages de réorganisation à ses autres contacts futurs (qui n'incrémenteront ainsi pas leur compteur). Cette dernière méthode, bien que plus simple, concentre toute l'information d'arrivée sur un seul participant (le nœud contacté), introduisant une perte totale en cas de départ de ce dernier. La première méthode de distribution de l'incrément est alors la plus robuste lorsque le réseau doit subir un dynamisme prononcé.

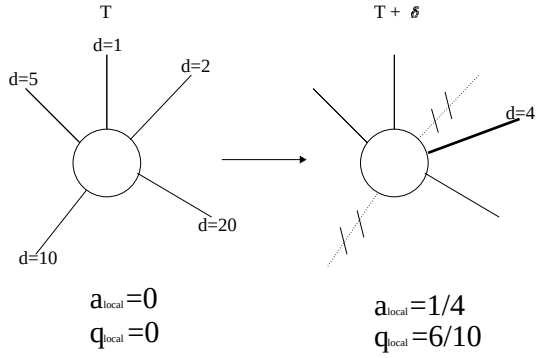

FIG. 2 – Évolution du voisinage d'un nœud et de ses compteurs de départs/arrivées après une période δ

Un exemple de ce mécanisme de surveillance est donné figure 2. Nous avons par exemple au temps $T + \delta$ deux détections de départs de voisins de degrés $d = 2$ et $d = 10$; nous obtenons alors simplement $q_{local} = \frac{1}{2} + \frac{1}{10} = \frac{6}{10}$.
A la fin de la phase de mesure locale, chaque nœud restant sur le réseau logique et ayant commencé l'observation δ unités de temps auparavant (nœud $\in \sigma$) possède une information locale de taux d'arrivées/départs sur le réseau logique. Afin d'homogénéiser ces mesures et de rendre l'information correcte à l'échelle du réseau, les nœuds de σ exécutent maintenant la seconde phase du protocole.

4.2. Phase d'agrégation pour évaluer le va-et-vient global

La seconde phase s'appuie sur un algorithme distribué de réduction de variance entre les nœuds participants ; nous cherchons donc à homogénéiser toutes les valeurs locales en une valeur globale sur le

réseau : $\bar{a} = \frac{1}{N}\Sigma_i^N a_{local\_i}$, pour les nœuds entrants et $\bar{q}$ pour les sortants.
Kempe et al. [12] proposent une approche dénommée *poussé/additionné* basée sur un mécanisme épidémique ; l'algorithme fonctionne avec des envois périodiques de couples poids/valeur, pour une convergence en au plus $O(\log(N) + \log(1/\epsilon) + \log(1/\delta))$ rondes (avec une probabilité d'au moins $1 - \delta$ et une erreur relative dans ϵ).
L'algorithme de Jelasity at al. [11] repose sur un mécanisme d'agrégation épidémique dit de *poussé/tiré* : à chaque intervalle de temps prédéfini (*ronde*), tout nœud du réseau choisit aléatoirement un autre nœud pour échanger sa valeur. Ils recalculent alors par exemple leur q_{local} local de la façon suivante : $q_{local} \leftarrow \frac{q_{local}+q_{contact}}{2}$. Dans un système sans erreur et non dynamique, cet algorithme tend à terme à l'homogénéisation de tous les q_{local}. Si un service sous-jacent de fourniture de nœuds sélectionnés de manière aléatoire et uniforme n'est pas disponible, cette sélection s'effectue parmi les voisins du nœud initiateur ; la variance des valeurs décroît alors, et ce quelle que soit la taille du réseau, de manière exponentielle en fonction du nombre de rondes de protocole écoulées (pour les réseaux comportant un degré d'aléatoire dans la construction de leur structure).

Algorithm 1 Mesure du dynamisme exécutée au nœud i

État du nœud i**:**
  a_{local_i}, le compteur d'arrivées, initialement 0,
  q_{local_i}, le compteur de départs, initialement 0,
  Δ, le temps entre l'envoi des messages de battements de cœur,
  k, le nombre de pertes de messages successives tolérées,
  V_i, initialement $\emptyset$, la vue contenant pour chaque voisin de i :
    j, son identifiant ;
    d_j, son degré ;
    $temporisateur_j$, le temps depuis la réception de son dernier message.

Phase de mesure:
  Périodiquement:
    for tout j voisin de i **do**
      if $temporisateur_j > \Delta k$ & !receptionMsgReorg **then**
        $q_{local\_i} \leftarrow q_{local\_i} + \frac{1}{d_j}$
        $V_i \leftarrow V_i \setminus \langle j, d_j \rangle$
        envoi d_i à V_i
  Sur connexion d'un nœud j**:**
    $V_i \leftarrow V_i \cup \langle j, d_j \rangle$
    envoi d_i à V_i
    if !receptionMsgReorg **then**
      $a_{local\_i} \leftarrow a_{local\_i} + \frac{1}{d_j}$

Phase d'agrégation:
  Périodiquement:
    envoi $\langle a_{local\_i}, q_{local\_i} \rangle$ à j //sélectionné aléatoirement et ayant participé à la phase 1
  Sur réception de $\langle a_{local\_j}, q_{local\_j} \rangle$ **de** j**:**
    envoiReponse $\langle a_{local\_j}, q_{local\_j} \rangle$ à i
    $a_{local\_i} \leftarrow (a_{local\_i} + a_{local\_j})/2$
    $q_{local\_i} \leftarrow (q_{local\_i} + q_{local\_j})/2$

L'algorithme dans son ensemble, avec l'heuristique de *poussé/tiré* [11] pour la phase d'agrégation, est résumé en Algorithme 1. Cette dernière heuristique signifie ainsi qu'à chaque cycle d'un nœud i, il envoie non seulement un message (poussé) à un voisin j mais en reçoit également un en réponse (tiré).
Nous avons présenté dans cette section un algorithme d'estimation des proportions de départs et d'arrivées des nœuds sur un réseau logique. Notons que ces informations sont disponibles à la fin de l'algorithme sur tous les nœuds présents sur le réseau au temps $T + \delta$, à l'exception des nœuds de l'ensemble

a (car arrivés en cours de protocole) ; ces derniers peuvent alors s'informer auprès des nœuds de σ. Nous évaluons maintenant notre méthode par simulation.

5. Évaluation de l'algorithme d'évaluation de va-et-vient

Notre algorithme a été simulé en utilisant PeerSim [18], un simulateur à évènements discrets dédié aux systèmes à grande échelle. Le but de cette simulation est de valider la justesse de la méthode ; des expériences intensives en environnement réel sont actuellement menées. Nous analysons ici les résultats de l'estimation du dynamisme du système. Le graphe créé est aléatoire [5] et connecté (méthode de construction PeerSim : $d_{min} = 1$, $d_{max} = 8$, $\bar{d} = 2$). Nous nous plaçons dans un cadre idéal, sans perte de messages, sans délais d'acheminement et à chaque ronde de protocole, les nœuds ont une connaissance correcte de l'évolution de leur voisinage. Ce type de topologie aléatoire rencontre depuis quelques années un succès grandissant, grâce entre autres à leur robustesse et à leur faible diamètre (voyez par exemple l'article général [17]). L'heuristique *poussé/tiré* a été employée dans les simulations suivantes, pour le calcul de la moyenne des mesures dans la seconde phase de l'algorithme.

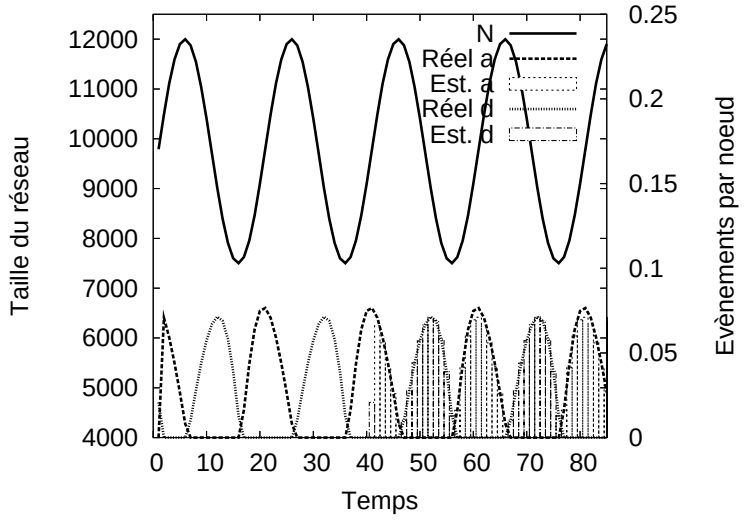

FIG. 3 – Qualité de l'estimation du dynamisme sur un réseau oscillant autour de $N = 10\,000$ nœuds

5.0.1. Réseau oscillant

La Figure 3 présente les résultats obtenus sur un réseau logique aléatoire de taille initiale $N = 10\,000$, oscillant par des départs et arrivées à $\pm 20\%$ de cette taille (courbe haute). Les suppressions et ajouts de nœuds se font également de manière aléatoire sur le réseau logique. L'axe des x représente le temps. L'axe y-gauche représente la taille du système comme étant le nombre de nœuds présents. L'axe y-droit représente quant à lui le taux réel d'arrivées et de départs ($\frac{card(a)}{N_T}$ et $\frac{card(q)}{N_T}$). Les deux courbes inférieures de la figure représentent les évolutions respectives du taux réel de départ et du taux réel d'arrivée. Les deux types de hachures verticales représentent les estimations respectives du taux de départ et du taux d'arrivée. Les résultats des estimations sont pris sur un nœud choisi aléatoirement. Nous pouvons constater que les premières estimations de notre méthode sont produites au temps 40, ce qui correspond au nombre de rondes (choisi arbitrairement ici) nécessaires pour obtenir au sein de chaque nœud la moyenne stabilisée des mesures effectuées dans la première phase. Par la suite les estimations sont fournies de manière permanente, car les deux phases peuvent s'exécuter en parallèle. La mesure est alors continue, une nouvelle débutant toutes les δ unités de temps, alors que l'agrégation est effectuée en tâche de fond, avec ici un décalage dans l'obtention du résultat d'un équivalent temps de 40 rondes. Pour témoigner de la justesse des mesures locales pour l'obtention du résultat final, la phase d'agrégation a eu lieu dans un contexte artificiel le préservant de dynamique. L'article [11] montre le faible impact des arrivées et départs de nœuds pendant l'exécution de l'algorithme de réduction de variance. En conclusion, cette figure montre que l'estimation des taux de départ et d'arrivée est correcte.

Départs	1%	5%	10%	20%	30%	40%	50%	60%	70%	80%	90%
Estimation	0,010	0,050	0,099	0,20	0,30	0,39	0,51	0,59	0,69	0,79	0,88

TAB. 1 – Résultats de l'estimation du dynamisme, en fonction du taux de départs aléatoires et simultanés pendant la période δ, sur un réseau de 10 000 nœuds

5.0.2. Départs massifs

Le tableau 1 présente quant à lui le cas habituellement problématique de départs massifs de nœuds du système. Nous considérons des retraits comme simultanés du point de vue de notre algorithme comme ayant lieu dans l'intervalle de temps dédié à la phase de mesure. Notre méthode confie à l'ensemble des nœuds du réseau logique l'information du dynamisme observé ; les nœuds qui quittent le réseau le font bien entendu avec leurs valeurs a_{local} et q_{local}. À la fin de la phase de mesure, si le système a connu des départs, l'information de dynamisme stockée sur le réseau est donc partielle. La phase de réduction de variance débute alors, et la moyenne effectuée l'est par les nœuds de l'ensemble σ. L'estimation obtenue à la fin de l'algorithme est alors $\frac{\widehat{card(q)}}{card(\sigma)}$ (où $\widehat{card(q)}$ représente l'information sur les départs subsistant sur le réseau à $T + \delta$) en lieu et place de la valeur exacte $\frac{card(q)}{N_T}$; il en va de même pour le taux d'arrivées. Les résultats présentés sur le tableau montrent cependant une très bonne précision de l'estimation fournie. Dans ce cas précis, l'intuition permettant de comprendre ce résultat surprenant est la suivante : comme l'information des départs/arrivées est distribuée de façon homogène sur le réseau (réseau aléatoire), si 50% des nœuds quittent le réseau logique, alors 50% de cette information va également disparaître. Le ratio $\frac{\widehat{card(q)}}{card(\sigma)}$ est alors constant, quel que soit le taux de départ, et fournit une bonne approximation du taux réel de départs et d'arrivées.

6. Discussions pratiques

6.1. Durée des phases

Afin d'obtenir un ordre d'idée dans la pratique du positionnement de la durée de mesure δ, nous pouvons la comparer à la durée non réductible de la seconde phase, celle d'agrégation basée sur un mécanisme épidémique. Si nous prenons pour base une valeur haute de 50 rondes de protocole avant la fin de cette phase (*cf* [11]), et que nous considérons que les nœuds du systèmes s'envoient des messages toutes les 5 secondes (propagation épidémique), nous obtenons une seconde phase d'à peu près quatre minutes. Pour conserver une certaine cohérence avec cette valeur, et également fournir des estimations sans un décalage trop grand entre le début de mesure et l'obtention du résultat final, il est souhaitable que le temps de mesure soit aussi de l'ordre de quelques minutes.

La caractérisation du dynamisme [22] sur des systèmes tels que Gnutella, Kad ou BitTorrent illustre très bien ceci. Voici le va-et-vient à attendre au sein des réseaux actuellement déployés : tous les pairs quittent le réseau après 30 minutes et sont remplacés par de nouveaux (hypothèse pessimiste, l'article [2] étudie le réseau Overnet et constate que chaque pair a rejoint et quitté le réseau en moyenne 6.4 fois par jour) ; si nous considérons une durée δ de mesure de 5 minutes, alors environ 15% des nœuds disparaissent pendant ce laps de temps.

6.2. Facteurs de dégradation de l'estimation

6.2.1. Inaccessibilités transitoires

L'article [16], traitant du déploiement de l'application ePOST sur la plateforme de test PlanetLab, souligne le problème des inaccessibilités transitoires de nœuds. Une fraction des machines surveillées deviennent temporairement inaccessibles de certaines parties du réseau lors de l'observation, alors qu'elles sont toujours physiquement présentes et qu'au moins une autre machine de l'ensemble des nœuds surveillés parvient à les contacter. Si nous voulons appliquer notre méthode à des réseaux à grande échelle basés sur IP, nous devons relacher l'hypothèse de fiabilité des liens de communications. Imaginons alors dans notre modèle un nœud i connecté à d_i voisins, dont un nœud j. Si le cas de figure évoqué se produit pour le couple i et j et pas pour les autres voisins de i, alors un biais est introduit dans notre méthode, pendant la phase de mesure locale. Le nœud j va incrémenter son compteur q_{local} de $1/d_i$, alors que i est toujours présent, violant ainsi l'invariant souhaité de $\Sigma_{i=1}^{N} q_{local\_i} = card(q)$. Il serait intéressant de voir, lors d'un déploiement, si ce phénomène se produit régulièrement pendant des phases de mesures relativement courtes et si l'impact sur l'estimation finale est significatif. Notons que si i décide de recréer

une connexion à cause de la perte de celle avec j, un message de réorganisation est envoyé au nouveau contact ; ce problème temporaire n'affecte donc pas l'estimation des arrivées.

6.2.2. Détecteur de défaillance

La précision de l'estimation locale, au cours de la première phase, dépend en partie du détecteur de défaillance qui sera employé. S'il nécessite un temps relativement important de détection d'erreur du processus voisin, par rapport à la durée de mesure δ, alors une partie non négligeable des nœuds ayant disparu pendant δ sera dénombrée lors de la mesure suivante (dans un processus de mesure continue). A l'inverse, le cas des arrivées est par nature plus réactif, puisqu'un nœud entrant s'annonce directement à un nœud désiré comme contact.

6.2.3. Algorithme de réduction de variance

Enfin, la qualité de l'estimation finale du dynamisme repose également fortement sur l'algorithme de réduction de variance employé dans la seconde phase de l'algorithme. Bien que les algorithmes épidémiques aient été simulés et étudiés formellement, et qu'une brève expérimentation ait été menée dans [11], il n'existe pas à notre connaissance de tests en déploiement effectif permettant de juger de leur robustesse et de leur précision en environnement réel. Une partie du travail futur consiste donc à implémenter notre méthode en environnement dynamique et à grande échelle.

6.3. Passage à l'échelle de la méthode

La première phase de l'algorithme est effectuée localement par chaque nœud. Chaque nœud échange un nombre constant de messages avec son voisinage et la taille de ce voisinage est fixée soit à une constante soit à $O(\log N)$ (avec une constante multiplicative raisonablement faible). Ainsi la complexité en temps et en communication nécessaire à la première phase de l'algorithme est très faible par rapport à N.
La seconde phase repose elle sur un algorithme basé sur un mécanisme épidémique, dont les propriétés de passage à l'échelle sont avérées [4]. En effet, l'article [11] montre à l'aide de simulations que le temps employé par l'algorithme de réduction de variance *poussé/tiré* est négligeable par rapport à N.
Le coût total de notre méthode pour une obtention sur chaque nœud de l'estimation est $O(N)$ (phase de mesure : $[coût\_détecteur\_défaillance] * 2E$, phase d'agrégation avec l'algorithme poussé/tiré : $2N * \#rondes$). La complexité en temps est de l'ordre de $O(\log N)$. L'algorithme d'estimation du va-et-vient passe ainsi à l'échelle.

7. Conclusion et perspectives

Cet article présente un algorithme distribué destiné à estimer l'intensité du dynamisme, appelée va-et-vient, d'un système à grande échelle. Cet algorithme permet aux nœuds participant au système d'estimer le taux de départs et le taux d'arrivées des nœuds ; ces informations peuvent alors être prises en compte à la volée par de nombreuses applications pour tolérer efficacement le dynamisme.
Notons qu'en plus des taux de départs/arrivées, il est possible d'estimer le nombre de nœuds partis ou arrivés : il suffit alors de multiplier l'estimation produite par notre algorithme par l'estimation de la taille du système [6, 1, 11] au temps T.
La simulation de notre solution sur des graphes aléatoires indique la justesse des évaluations obtenues en systèmes à grande échelle. Nous avons récemment développé *GossiPeer* [8] permettant le déploiement à grande échelle de protocoles épidémiques. Afin de généraliser nos résultats obtenus en environement synchrone, cet algorithme est en cours d'adaptation sur *GossiPeer* afin d'être déployé à grande échelle sur un ensemble de machines distribuées géographiquement. Une direction de recherche intéressante est d'évaluer l'impact que peut avoir, sur la précision des estimations, la latence des messages dans un environement ouvert.
La recherche d'algorithmes fournissant des informations clés sur le système, dans un contexte à grande échelle et dynamique, fait actuellement l'objet d'une attention importante. Elle s'inscrit dans l'amélioration de la qualité de service rendue par l'application distribuée, et ce par un mécanisme de contrôle/réadaptation permanent à l'environnement d'exécution. Nous pensons qu'une adaptation de notre algorithme peut fournir d'autres statistiques intéressantes sur le système, comme par exemple des informations concernant la distribution des temps de session des nœuds.

Bibliographie

1. M. Bawa, H. Garcia-Molina, A. Gionis, et R. Motwani. Estimating aggregates on a peer-to-peer network. Technical Report, Dept. of computer science, Stanford University, 2003.
2. R. Bhagwan, S. Savage, et G. M. Voelker. Understanding availability. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems*, volume 2735 of *Lecture Notes in Computer Science*, pages 256–267. Springer Berlin / Heidelberg, February 2003.
3. Andreas Binzenhöfer et Kenji Leibnitz. *Managing Traffic Performance in Converged Networks (chapter : Estimating Churn in Structured P2P Networks)*. Springer Berlin / Heidelberg, New York, NY, USA, 2007.
4. A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, et D. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12, New York, NY, USA, 1987. ACM Press.
5. P. Erdös et A. Rényi. On random graphs. In *Publicationes Mathematicae*, pages 6 : 290–297, 1959.
6. A. J. Ganesh, A.-M. Kermarrec, E. Le Merrer, et L. Massoulié. Peer counting and sampling in overlay networks based on random walks. *To appear in Distributed Computing journal*, 2007.
7. G. Ghinita et Y. M. Teo. An adaptive stabilization framework for distributed hash tables. *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 25–29, April 2006.
8. V. Gramoli, A.-M. Kermarrec, et E. Le Merrer. GossiPeer : `http ://gossipeer.gforge.inria.fr`.
9. V. Gramoli, A.-M. Kermarrec, A. Mostéfaoui, M. Raynal, et B. Sericola. Core persistence in peer-to-peer systems : Relating size to lifetime. In *Proceedings of the On The Move International Workshop on Reliability in Decentralized Distributed Systems*, volume 4278 of *LNCS*, pages 1470–1479. Springer, April 2006.
10. Vincent Gramoli, Anne-Marie Kermarrec, Achour Mostefaoui, Michel Raynal, et Bruno Sericola. Persistance de noyau dans les systèmes dynamiques à grande échelle. In *9ème Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, pages 59–62, 2007.
11. M. Jelasity, A. Montresor, et O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3) :219–252, 2005.
12. D. Kempe, A. Dobra, et J. Gehrke. Gossip-based computation of aggregate information. In *In Proc. of the 44th Annual IEEE Symposium on Foundations of Computer Science*, 2003.
13. F. Kuhn, S. Schmid, et R. Wattenhofer. A self-repairing peer-to-peer system resilient to dynamic adversarial churn. In *Proc. 4th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2005.
14. X. Liu, L. Xiao, A. Kreling, et Y. Liu. Optimizing overlay topology by reducing cut vertices. In *NOSSDAV 2006 : International Workshop on Network and Operating Systems Support for Digital Audio and Video*, Newport, Rhode Island, USA, 2006.
15. L. Massoulié, A.-J. Ganesh, et A.-M. Kermarrec. Network awareness and failure resilience in self-organising overlay networks. In *IEEE Symposium on Reliable and Distributed Systems*, 2003.
16. A. Mislove, Post A, A. Haeberlen, et P. Druschel. Experiences in building and operating epost, a reliable peer-to-peer application. *SIGOPS Oper. Syst. Rev.*, 40(4) :147–159, 2006.
17. M. E. J. Newman. Random graphs as models of networks, Feb 2002.
18. PeerSim : `http ://peersim.sourceforge.net`.
19. B. Porter, F. Taiani, et G. Coulson. Generalised repair for overlay networks. In *SRDS '06 : Proceedings of the 25th IEEE Symposium on Reliable Distributed Systems (SRDS'06)*, pages 132–142, Washington, DC, USA, 2006. IEEE Computer Society.
20. T. Qiu, E. Chan, et G. Chen. Overlay partition : Iterative detection and proactive recovery. In *ICC2007 : The IEEE International Conference on Communication*, Glasgow, Scotland, 2007. IEEE Computer Society.
21. A. Rowstron et P. Druschel. Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218, 2001.
22. D. Stutzbach et R. Rejaie. Understanding churn in peer-to-peer networks. In *IMC '06 : Proceedings of the 6th ACM SIGCOMM on Internet measurement*, pages 189–202, New York, NY, USA, 2006.
23. J. Tian et Y. Dai. Understanding the dynamic of peer-to-peer systems. In *IPTPS 2007 - Sixth International Workshop on Peer-to-Peer Systems*, Bellevue, USA, 2007.