HAL
open science

Evaluating topology quality through random walks

Anne-Marie Kermarrec, Erwan Le Merrer, Bruno Sericola, Gilles Trédan

To cite this version:

Anne-Marie Kermarrec, Erwan Le Merrer, Bruno Sericola, Gilles Trédan. Evaluating topology quality through random walks. [Research Report] RR-6534, INRIA. 2008, pp.22. inria-00279074v2

HAL Id: inria-00279074

https://inria.hal.science/inria-00279074v2

Submitted on 15 May 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Evaluation de la qualité d'une topologie réseau grâce à des marches aléatoires

Anne-Marie Kermarrec — Erwan Le Merrer — Bruno Séricola — Gilles Trédan

N° 6534

Mai 2008

Thème NUM

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--6534--FR+ENG

Evaluation de la qualité d'une topologie réseau grâce à des marches aléatoires

Anne-Marie Kermarrec, Erwan Le Merrer, Bruno Séricola, Gilles Trédan

Thème NUM — Systèmes numériques
Équipes-Projets ASAP et ARMOR

Rapport de recherche n° 6534 — Mai 2008 — VII pages

Résumé : Un système ou un réseau distribué peut être modélisé comme un graphe représentant la relation "qui connait qui". La *conductance* d'un graphe exprime la qualité de sa connectivité. Dans un réseau composé de larges et denses clusters, connectés par seulement quelques liens, le risque de partitionnement est important; ceci est typiquement reflété par la notion de conductance. Calculer cette conductance pour un graphe est une tâche coûteuse et complexe, qui requiers la connaissance de la structure complète de la topologie qui le caractérise.

Au delà de l'information portée par la conductance, ce qui compte réellement est d'identifier les nœuds critiques du point de vue topologique. Nous proposons dans ce rapport un algorithme totalement distribué qui se propose de fournir à chaque nœud une valeur qui reflète la qualité de sa connectivité au sein du graphe. Comparer ces valeurs entre nœuds permet d'avoir une approximation locale des caractéristiques globales d'un graphe. Notre algorithme repose sur une sonde anonyme qui visite le réseau d'une manière non biaisée par la diversité des degrés des nœuds du graphe. Chaque nœud enregistre les temps écoulés entre les visites de la sonde (appelés les *temps de retour*). Calculer l'écart type de tels temps de retour sur chaque nœud permet de fournir une information qui peut être employée pour déterminer leur position relative dans la topologie, déduisant alors le fait qu'il soient plus ou moins critiques. Des algorithmes d'amélioration de la connectivité du graphe peuvent être alors déclenchés. Les moments d'ordre 1 et 2 des temps de retour sont étudiés analytiquement en utilisant le modèle des chaines de Markov, et montre que les temps de retour sont effectivement liés à la position des nœuds sur le graphe. Nous évaluons ensuite notre algorithme à l'aide de simulations. Les résultats produits montrent que notre algorithme est capable de fournir des indicateurs correlés à la conductance d'un graphe donné. Nous pouvons par exemple détecter précisément les nœuds *ponts* dans une topologie constituée de deux denses clusters connectés simplement par un lien.

Mots-clés : Marches aléatoires non biaisées à temps discret, Temps de retour, Conductance, Détection de goulots d'étranglement

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex
Téléphone : +33 2 99 84 71 00 —Télécopie : +33 2 99 84 71 71

Evaluating the quality of a network topology through random walks

Abstract: A distributed system or network can be modeled as a graph representing the "who knows who" relationship. The *conductance* of a graph expresses the quality of the connectivity. In a network composed of large dense clusters, connected through only a few links, the risk of partitioning is high; this is typically reflected by a low conductance of the graph. Computing the conductance of a graph is a complex and cumbersome task. Basically, it requires the full knowledge of the graph and is prohibitively expensive computation-wise.

Beyond the information carried by the conductance of a graph, what really matters is to identify critical nodes from the topology point of view. In this paper we propose a fully decentralized algorithm to provide each node with a value reflecting its connectivity quality. Comparing these values between nodes, enables to have a local approximation of a global characteristic of the graph. Our algorithm relies on an anonymous probe visiting the network in a unbiased random fashion. Each node records the time elapsed between visits of the probe (called return time in the sequel). Computing the standard deviation of such return times enables to give an information to all system nodes, information that may be used by those nodes to assess their relative position, and therefore the fact that they are critical, in a graph exhibiting low conductance. Based on this information, graph improvement algorithms may be triggered. Moments of order 1 and 2 of the return times are evaluated analytically using a Markov chain model, showing that standard deviation of return time is related to the position of nodes in the graph. We evaluated our algorithm through simulations. Results show that our algorithm is able give informations that are correlated to the conductance of the graph. For example we were able to precisely detect bridges in a network composed of two dense clusters connected through a single link.

Key-words: Unbiased Discrete Time Random Walks, Return times, Conductance, Bottlenecks Detection

1 Motivations

This section presents the goal of this work. In this paper we propose a distributed algorithm to assess the connectivity quality of a network, be it physical or logical. We show that in large-scale networks, both local (from a node viewpoint) and global (from an administrator viewpoint) decisions might benefit from such an algorithm. The knowledge of an issue in the network layout is effectively the first step toward its effective repair.

Large scale peer-to-peer, grids or wireless sensors networks often present a really complex and huge interaction graph structure. Such graphs usually contain such a large amount of information that it is extremely difficult to process any information aggregated from all nodes.

Graph theory provides indicators of the global graph layout (*e.g. :* conductance, spectral gap). Computing these indicators is expensive, usually relies on centralized solutions, and requires a total knowledge of the graph. This almost precludes their use in a distributed large scale system. In the absence of practical solutions, it is extremely difficult to assess global properties on a graph topology in a given network. Yet, we introduce the notion of *health* of a graph : a healthy graph is a graph that fits the needs of its upperlayer application. The health of such an interaction graph is crucial for most applications exploiting it. Network partition is one of the most feared issues. Some applications also rely on uniform graph hypotheses, which is an assumption that may even fail more frequently than partitioning occurs.

Networks are sensitive to failures that may severely impact the general layout of the graph : massive disconnections due to ISP failures may lead to correlated link failures that bias the connectivity and uniformity of graphs. Failures can also be due human decisions, which are inherently unpredictable : interests are not uniform [Adl], and the resulting dynamics may also impact a graph. Moreover, these problems are likely to get worse over time : biased peer selection or bridges congestion may turn a bridged graph into a partitioned. clear for everyone at this point ?

Ad-hoc network literature provides graph partitions detection methods (*e.g.* [SLS06]), but these methods are either specific to planar graphs, either based on k-hop neighborhood knowledge that is not very realistic in large-scale systems. In [MPHD06], the authors show that even a highly structured and controlled overlay such as Pastry, a large-scale distributed hash table, is often subject to partitions in practice. In all these situations, the knowledge of potential regions of risks can be extremely helpful to prevent partitions.

Conductance is a useful health indicator that, in a nutshell, captures the weakest part of the graph. Weakest here refers to a the smallest set of links that connect the largest set of nodes to the rest of the graph. Section 3 provides a formal definition of conductance. Conductance may have an impact on several applications such as peer sampling algorithms or system size estimations (see e.g. paper [MMKG06] that shows that the conductance parameter of the graph directly impacts the quality of the estimator). Providing nodes with such an indicator would allow them to estimate the quality of the results provided by these applications.

From a global viewpoint, a system administrator can exploit graph health indicators to prevent disconnections and partitions and more generally to issue statistics about the interaction graph. This can also be the first step toward

identifying clusters and issuing repair. Repair can in itself be improved by such a method, since it can be triggered only on problematic nodes. Moreover, indicators such as conductance are handy tools to detect causes of graph degeneration.

In this paper, we propose a fully decentralized approach resulting on labelising nodes with a value reflecting their relative "importance" in the communication network, so that distributed detection mechanism can be triggered to preventively repair or improve the network. The idea is to use a probe that randomly walks the whole system in an unbiased fashion. By observing this probe's return times, and the standard deviation of these, nodes infer valuable data about the graph health. Nodes are both able to detect *1)* whether the graph is healthy or not and *2)* their own criticity w.r.t. this health status.

The following is a brief roadmap of this paper : Section 2 describes the theoretical model our algorithm is based upon. Section 3 presents related problems from the graph theory standpoint. The intuition of our algorithm and the algorithm itself are presented in section 4. In section 6.1, we analyse the standard deviation of return times and show that it is related to the node's position in the graph. To assess he validity of our approch, we conducted some simulations. The results are presented in Section 6. We also present the accuracy of our algorithm in detecting bridges. In Section 7, we discuss the distributed exploitation and the convergence time of our approach before concluding.

2 System model

We consider a n node connected network, represented as an undirected graph $\mathcal{G} = (V, E)$, with n vertices and m edges. For a node $i \in V$, Γ_i denotes the set of neighbors of i in $\mathcal{G}$ (vertices with an adjacent edge to i), and d_i its degree, namely the size of Γ_i.

Note that this graph may represent any peer-to-peer, grid, social, or physical network.

We use a *probe*, in other words a *random walk*, on the graph $\mathcal{G}$, *i.e.* a process progressing in the network from a node i to another node chosen at random in i's neighborhood Γ_i. We consider this walk to be *permanent* for it has no stop condition. This is one of the main differences with most of random walk based algorithms (e.g. sampling [MMKG06], search [LCC+02]). We assume that the probe is initiated by one node of the network. We also assume, for the sake of comprehension, that the network is static, *i.e.* the topology does not change during the algorithm execution. The latter assumption ensures that the random walk never stops.

Note that our model only requires one probe for the whole system, as opposed for example to one probe launched by each node. Thus a single probe can be leveraged by the whole set of nodes.

3 Related work

In this section, we survey the works related to connectivity problems of networks.

Min-cut A first related problem to derive bottleneck problems in a graph is named the *min-cut* problem. A *cut* in a graph $\mathcal{G}$ is a partition of the vertices $V = (C, \bar{C})$, with $\bar{C} = V \setminus C$ and with C and $\bar{C}$ being two non-empty sets. The min-cut is then the minimal number of edges that cross a cut C to $\bar{C}$. Book [MR95] presents a randomized algorithm that computes a min-cut with probability $\Omega(1/\log n)$ and runs in $O(n^2 \log n)$ time. A deterministic approach [SW97] for weighted graphs gives a min-cut in $O(m + nlogn)$. Unfortunately those solutions remain purely centralized and require a full knowledge of the system graph to give as an output the edges implied in the min-cut.

Spectral gap and conductance The notions of *spectral gap*, λ_2, and *conductance*, Φ, are also tightly related to the connectivity issues of a graph. The value λ_2 is given by finding the Eigenvalues of the Laplacian matrix of $\mathcal{G}$, L ($L = D - A$, with D and A respectively the diagonal matrix of degrees and the adjacency matrix of $\mathcal{G}$), and sort them as $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$. Φ is defined by

$$\Phi := \inf_{C:|C| \leq N/2} \frac{E(C, \overline{C})}{|C|},$$

where $E(C, \overline{C})$ denotes the number of edges in the graph $\mathcal{G}$ between the set of nodes C, and the complementary set $\overline{C}$ (see e.g.[Moh97]). Finally, those two values λ_2 and Φ are linked by the Cheeger inequality that states that $\lambda_2 \geq \frac{\Phi^2}{2\Delta(\mathcal{G})}$ [Moh97], where $\Delta(\mathcal{G})$ is the maximal degree of nodes in the graph. Then the lowest the values λ_2 and Φ are, the higher the probability that algorithms that are run on top of the network behaves poorly, and/or slowly. For example, paper [MAS06] shows how conductance directly affects the running time of a gossip-based algorithm that computes separable functions. Unfortunatly, we are unaware of methods to distribute the computation of both values.

Paper [Sin92] also relates Markov chain mixing times with spectral gap, conductance and network flow problems.

A distributed approach On a more practical point of view, a recent paper [JFDG07] shows that on a P2P network using *network coding*, linear combinations a peer receives from its neighborhood could be passively used to infer topology bottlenecks. We target an algorithm that could be ran on top of every network rather than content distribution oriented networks.

4 Perpetual random walks

4.1 Intuition

4.1.1 The high clustering intuition

To illustrate our purpose, we voluntary consider an extreme setting, known as the *barbell graph*. The following barbell graph is composed of two complete components of m_1 nodes each, connected by a path of m_2 nodes.

Figure 1 presents an example of size $m_1 = 5$ and $m_2 = 2$. It can easily be shown (see e.g. [AF]) that a random walk starting at any node in the "left bell", say v_l, takes as mean time $m_1^2 m_2$ steps to reach another node, v_r, in the "right bell" (and conversely). On the other hand, nodes from the left bell *see*

the random walk really often (with probability $1/m_1$ at each step) before it crosses the path to the other bell ; those nodes are then able to observe high standard deviation between the return times of the random walk, depending on the passage or not to the right bell. By simply comparing the return times of the walk, every graph node can then locally detect the presence of "traps" on the topology.

More accurate assessment of the network can be made from the observation of those return times. Consider the node v_L (resp. v_R) : it has the same probability that the nodes in its bell to see the random walk at each step ; but once the walk has passed to the right bell (resp. left bell), this node is the mandatory passage point for the walk to get back into the other bell. It turns out that those bridge nodes see the walk more regularly, and then observe reduced standard deviation over the return times. Now assume that each graph node communicates periodically with its direct neighbors by exchanging and comparing the deltas of their return times, then bridge nodes can discover that they have a particular position in the graph.

4.1.2 The electric lattice intuition

Random walks could also be used to provide characteristics of electrical materials. A method by Doyle and Snell [DS00] enables to find the effective electrical resistance of a composite. In a nutshell (see [PC03]), a discrete representation of the material is created using a lattice whose edges have resistances determined by the underlying material. A set of nodes I from that lattice represents the side through which the current enters, and another set O where it leaves. One of the edges connected to I is chosen at random so that a random walk starts ; it ends either in reaching a node of O or re-reaches I. The probability that the walk gets to O before returning to I (so called *escape probability*) is shown to be proportional to the conductance of the lattice. Composite conductance is then tightly related to graph bottlenecks issues, and that notion of sticky walk in a composite, in a cluster in our case, is at the core of our approach.

4.2 The permanent probe algorithm

The algorithm we propose is very simple : each graph node logs and computes the standard deviation of the return times of a permanent unbiased random walk, or probe, running on the topology. This information can then be used to assess *health* of the graph.

4.2.1 Unbiased random walk

Our algorithm uses a unbiased random walk, namely a walk whose *stationary distribution* π_i, for all $i \in V$, is $1/n$. A biased (or simple) random walk, puts mass on high degree nodes, as $\pi_i = d_i/2m$ (see e.g. [MR95]). We unbias the random walk by using the Metropolis-Hastings method [Has70, SRD+06] ; each step of the walk from some node i proceeds as follows :

The aim of unbiasing the random walk is to ensure that the only cause of the variations on return times on nodes is due to conductance issue. this is achieved by removing the degree bias for non regular graphs. Moreover, as it ensures

```
Choose a neighbor j from Γ_i uniformly at random
Query j for d_j (its current degree)
Generate a random number p ∈ [0, 1] uniformly
if p ≤ d_i/d_j then
    make the step to j
else
    remain at i
```

that all nodes eventually see the probe an equal number of times, it fastens the convergence by evenly providing return times to all nodes.

4.2.2 Standard deviation of return times

The key point of the algorithm is the variations of the frequency at which a random walk visits system nodes.

Every node i in $\mathcal{G}$ joins the detection process on the first passage of the walk, by creating an array Ξ_i that logs every return time. A simple solution to capture irregularity of visits on nodes is to proceed as follows : after the third return, a node i computes the standard deviation σ_i of the *return times* recorded in Ξ_i (*i.e.* the time needed for the walk, starting at i, to return to i).

Return times on a particular node can be either absolute time, thus implying a clock on every system node, or simply the number of steps proceeded by the random walk (which thus carries that information across the network). Note that in the first case, nodes' clocks do not need to be synchronized, as we are only interested in local standard deviation of the return times.

5 Return times in Markov chains

We now formaly prove that the return time of a random walk on a particular node is function of the position of this node in a given graph. This idea is at the core of our approach, and is generally forgotten due to the fact that litterature often provides bounds to return time probability laws that hide the local variations nodes may experience (big O notation). Those potentially small variations may suffice to differentiate nodes with respect to their position.

States of the Markov chain are vertices E of $\mathcal{G}$. The general case of a biased walk is presented here ; an unbiased walk is simply a subcase of it. Proofs are deferred in the appendix.

Let $X = \{X_n,\ n \in \mathbb{N}\}$ be a homogeneous and irreducible discrete time Markov chain on the finite state space S. We denote by $P = (P(i,j))_{i,j\in S}$ its transition probability matrix and we are interested in the computation of the return times for every state of S. For every state in S, we denote by $\tau(j)$ the number of transitions needed to reach state j, i.e.

$$\tau(j) = \inf\{n \geq 1 \mid X_n = j\}.$$

The state space S being finite and X being irreducible, X is recurrent which means that $\tau(j)$ is finite a.s. We denote by $f_j^{(n)}(i)$ the distribution of $\tau(j)$ when the initial state of X is i, that is, for every $n \geq 1$,

$$f_j^{(n)}(i) = \mathbb{P}\{\tau(j) = n \mid X_0 = i\}.$$

$f_i^{(n)}(i)$ represents the probability, starting from state i, that the first return to state i occurs at instant n and, for $i \neq j$, $f_j^{(n)}(i)$ represents the probability, starting from state i, that the first visit to state j occurs at instant n. These probabilities are given by the following theorem.

Theorem 1 *For every $i, j \in S$ and $n \geq 1$, we have*

$$f_j^{(n)}(i) = \begin{cases} P(i,j) & \text{if} \quad n = 1 \\ \displaystyle\sum_{\ell \in S - \{j\}} P(i,\ell) f_j^{(n-1)}(\ell) & \text{if} \quad n \geq 2. \end{cases} \tag{1}$$

For every $j \in S$ and $n \geq 1$, we denote by $f_j^{(n)}$ the column vector containing the values $f_j^{(n)}(i)$ for every $i \in S$. For every $j \in S$, we introduce the matrix Q_j obtained from matrix P by replacing the jth column by zeros, that is

$$Q_j(i,\ell) = \begin{cases} P(i,\ell) & \text{if} \quad \ell \neq j \\ 0 & \text{if} \quad \ell = j. \end{cases}$$

We also introduce the column vector P_j containing the jth column of matrix P, i.e. $P_j(i) = P(i,j)$. Equation (1) can then be written in matrix notation as

$$f_j^{(n)} = \begin{cases} P_j & \text{if} \quad n = 1 \\ Q_j f_j^{(n-1)} & \text{if} \quad n \geq 2, \end{cases} \tag{2}$$

which leads to an easy computation of the vectors $f_j^{(n)}$. We now define the matrix $M = (M(i,j))_{i,j \in S}$ by $M(i,j) = \mathbb{E}\{\tau(j) \mid X_0 = i\}$. $M(i,i)$ represents the expected time between two successive visits of X to state i, and, for $i \neq j$, $M(i,j)$ represents the expected time, starting from state i, to reach state j for the first time. The Markov chain X being irreducible, we have $M(i,j) < \infty$ for every $i, j \in S$ and

$$M(i,i) = \frac{1}{\pi_i},$$

where π_i is the ith entry of the probability distribution π, which is the unique solution to the system $\pi = \pi P$.

Note that when the graph of the Markov chain is unbiased, i.e. when $P(i,j)$ is the same for every pair (i,j) such that $i \neq j$, the matrix P is bistochastic and thus $\pi_i = 1/|S|$ and $M(i,i) = |S|$. For what concerns the second order moments $H(i,i)$ of the return times to state i, we have from Corollary 3,

$$(I - Q_j)H_j = (I + Q_j)M_j.$$

If matrix P is bistochastic, we have $\mathbb{1}^t P = \mathbb{1}^t$, where t denotes the transpose operator and thus $\mathbb{1}^t Q_j = \mathbb{1}^t - e_j$, where e_j is the jth unit row vector, i.e. $e_j(i) = 1$ if $i = j$ and 0 otherwise. Multiplying each side of the previous relation by vector $\mathbb{1}^t$, we get

$$\mathbb{1}^t H_j - \mathbb{1}^t Q_j H_j = \mathbb{1}^t M_j + \mathbb{1}^t Q_j M_j,$$

which simplifies as

$$\mathbb{1}^t H_j - (\mathbb{1}^t - e_j)H_j = \mathbb{1}^t M_j + (\mathbb{1}^t - e_j)M_j$$

and thus

$$H(j,j) = 2\sum_{i\in S} M(i,j) - M(j,j) = 2\sum_{i\in S} M(i,j) - |S|.$$

The standard deviation $\sigma(j)$ then writes

$$\sigma(j) = \sqrt{2\sum_{i\in S} M(i,j) - |S|(|S|+1)}.$$

To compute all the entries of matrix M, we need the following result. We introduce the column vector M_j containing the jth column of matrix M, i.e. $M_j(i) = M(i,j)$ and the column vector of ones denoted by $\mathbb{1}$. These expected values are given by the following result.

Corollary 2 *For every $j \in S$, we have*

$$M_j = (I - Q_j)^{-1}\mathbb{1}.$$

In practice, the column vector M_j is obtained for every $j \in S$ by solving the linear system $(I - Q_j)M_j = \mathbb{1}$.

Let us consider now the second moment of $\tau(j)$. We define the matrix $H = (H(i,j))_{i,j\in S}$ by $H(i,j) = \mathbb{E}\{\tau(j)^2 \mid X_0 = i\}$. $H(i,i)$ represents the second moment of the time between two successive visits of X to state i, and, for $i \neq j$, $H(i,j)$ represents the second moment of the time, starting from state i, to reach state j for the first time. We introduce the column vector H_j containing the jth column of matrix M, i.e. $H_j(i) = H(i,j)$. These values are given by the following result.

Corollary 3 *For every $j \in S$, we have*

$$H_j = (I - Q_j)^{-1}(I + Q_j)M_j.$$

In practice, the column vector H_j is obtained for every $j \in S$ by solving the linear system $(I - Q_j)H_j = (I + Q_j)M_j$.

The standard deviation $\sigma(i)$ of the return time to state i is thus given by

$$\sigma(i) = \sqrt{H(i,i) - [M(i,i)]^2}.$$

Note that when the graph of the Markov chain is unbiased, i.e. when $P(i,j)$ is the same for every pair (i,j) such that $i \neq j$, the matrix P is bistochastic and thus $\pi_i = 1/|S|$ and $M(i,i) = |S|$. For what concerns the second order moments $H(i,i)$ of the return times to state i, we have from Corollary 3,

$$(I - Q_j)H_j = (I + Q_j)M_j.$$

If matrix P is bistochastic, we have $\mathbb{1}^t P = \mathbb{1}^t$, where t denotes the transpose operator and thus $\mathbb{1}^t Q_j = \mathbb{1}^t - e_j$, where e_j is the jth unit row vector, i.e. $e_j(i) = 1$ if $i = j$ and 0 otherwise. Multiplying each side of the previous relation by vector $\mathbb{1}^t$, we get

$$\mathbb{1}^t H_j - \mathbb{1}^t Q_j H_j = \mathbb{1}^t M_j + \mathbb{1}^t Q_j M_j,$$

which simplifies as

$$\mathbb{1}^t H_j - (\mathbb{1}^t - e_j)H_j = \mathbb{1}^t M_j + (\mathbb{1}^t - e_j)M_j$$

and thus

$$H(j,j) = 2\sum_{i\in S} M(i,j) - M(j,j) = 2\sum_{i\in S} M(i,j) - |S|.$$

The standard deviation $\sigma(j)$ then writes

$$\sigma(j) = \sqrt{2\sum_{i\in S} M(i,j) - |S|(|S|+1)}.$$

6 Evaluation

6.1 Execution on a micro example

As we mentioned before, computing the conductance of a graph is extremely costly. Therefore, in order to compare the values obtained with our algorithm against real values, we considered a a small setting of 10 nodes in order to be able to compute the conductance values. Figure 2 plots a typical run of the algorithm, with the standard deviation of probe visits on every node (italic values). The blue bold values are the theorical values of standard deviation provided in section .The conductance Φ has been computed by hand for every adjacent vertices of this small example ; $\Phi = 0.2$, we also present the 2 other smallest values and the cut implied (dashed lines).

Nodes at the edges of the graph have a relatively high σ, and nodes in the middle of $\mathcal{G}$ the lowest value, following the fact that the return time of the random walk on nodes at the edges is higher than for centered nodes, on that graph close to a *line* or *path graph* ; theorical return times can be easily computed with tools provided in Section 6.1. We observe that the importance of nodes in the topology is effectively correlated to the inverse of their value order : smallest σ values are effectively related to the smallest conductance values.

6.2 Simulations on pathologic graphs

Graph construction We consider random graphs, that are constructed on the Erdös-Rényi model [ER59], with the probability parameter that two edges of $\mathcal{G}$ are connected given by $p = \frac{2\ln n}{n}$. Probability $p > \frac{\ln n}{n}$ insures connectivity of $\mathcal{G}$, *i.e.* that no vertex is isolated ; n varies in our experiments. Random graphs represent a typical target topology for many applications, as an example of a "healthy" connectivity, that provides high resilience to failures, low clustering, and a small diameter [New02].

Clusterized graphs are simply composed of two random components, linked by x edges (as on Figure 1, where $x = 1$).

Distribution of standard deviations Figure 3 presents an histogram of the distribution of standard deviations on each node, after 5.10^5 steps of the probe. Parameters are set as follow : $n = 2 \times 250$ and $x = 1$.

The thiner the spike, the more homogeneously the σ values are distributed on the network, as a result of our algorithm. The left spike is related to the the random graph simulations while the right one is related to the clusterized ones. This latter one is concentrated around 4 times the average value for the random graph, indicating that visits of the probe on the nodes are far more irregular, due to the topology characteristics.

We also observe that the tail of the clusterized graph's curve is longer showing discrepancies between nodes in the same graph : some nodes see the probe more regularly than others. This observation leads us to look at a more precise example of this σ values' repartition.

Figure 5 provides a visual example of the distribution of those values on all system nodes. The considered network, Figure 5(a), is a 2-Dimensional network, on the model of a wireless sensor network for example, where nodes are connected to their close geographic neighbors. On Figure 5(b), darker zones correspond to low standard deviations while clearer zones corresponds to higher values of σ. It is clear from this figure that nodes at the edges of the topology are visited irregularly, and nodes on the bridges have low values, as they are mandatory passage points. Those obvious differences may be used to detect, in a distributed fashion, those central, potentially critical, nodes by comparing such values.

An animation of the evolution of the σ values on nodes is available at [vid] ; snapshots of the network are taken every 25.10^3 steps.

Bridges detection We now look at potentially critical nodes of the topology : the bridges. Those nodes are likely to receive a high pressure from applications that are run on top of the topology, due to their strategical positions (*e.g.* routing or flooding algorithms).

Settings of the experiment being $n = 2 \times 500$ and $x = 1$, we focus on the σ values of the two nodes sharing the edge that gathers the two random graphs. Those two bridge nodes are known in our simulations, so that we look at the position of their σ compared to those of all network nodes. In other words, we check out if their values are effectively among the smallest values in the system. Figure 6 depicts the effective position of those bridges in either 1 or 5% of the smallest σ values of all nodes in $\mathcal{G}$ as a function of the number of probe steps. Results shown are the average of the results of 20 experiments. We observe a convergence of the algorithm toward an accurate ordering.

It is important to notice that our experiments have been conducted with just one bridge between graphs, clearly providing a worst case scenario for the convergence time. By considering that previous experiments give satisfying results and match the intuitions, we then conclude that our method is efficient to give an indicator of the relative importance of nodes in the topology.

7 Discussion

We now discuss some critical points.

7.1 Algorithm convergence time

As our algorithm computes on each node the standard deviations of the probe's return times, its convergence time (*i.e.* each node has a representative σ value) is related to the *cover time* of $\mathcal{G}$. Cover time is the number of steps needed by a random walk to visit each vertex of $\mathcal{G}$. U. Feige [Fei95] showed that for any undirected graph, expected cover time is at most $\frac{4}{27}n^3 + o(n^3)$. This upper bound is known for the extreme case of a *lollipop* graph (a clique of $\frac{n}{2}$ nodes, linked to a path of the remaining nodes) ; less severe graph degeneration then leads to far reduced cover times. As we obviously need more than one visit by node to compute σ, say k visits, when then have this extra factor, times the worst case cover time to reach convergence.

We oppose to that high convergence time the fact that no other method is known to highlight conductance problems in a distributed fashion. Computing conductance is a NP-hard problem. Our approach is then a good candidate to be used permanently as a background monitoring mechanism.

7.2 Distributed detection

Depending on the application, a target standard deviation σ_{ideal} may be provided to each node so that it is able to estimate its position compared to the ideal situation captured by σ_{ideal}. Computing its own standard deviation σ, a node can compute the ratio $\frac{\sigma}{\sigma_{ideal}}$, triggering repair if this ratio increases over a predefined threshold, in a totally distributed fashion. σ_{ideal} can be computed using tools provided in section 6.1, or through simulations.

Nodes may also exchange some of their probes passage times. Suppose nodes a and b, exchange their sets of passage times Ξ_a and Ξ_b. The ratio $r_{a\to b} = \frac{\sigma(\Xi_a \cup \Xi_b)}{\sigma(\Xi_a)}$ can be exploited as a distributed cluster detector : if a and b are located in two different clusters, then the standard deviation of the union of passage dates is small, so that $r_{a\to b}$ is low. On the other side, if nodes a and b are in the same cluster, the probe is likely to hit both at very close periods, so that $r_{a\to b}$ converges to 1. a and b are thus both able to identify their respective position in a fully decentralized way.

7.3 Conclusion and future work

In this paper, we were interested to assess in a distributed way the *health* of a graph. We showed that a single random walk, run permanently in the system, can provide the system nodes with their own criticity. with respect to the health of the topology. This constitutes a major indicator for a graph topology since it both reflects the global interaction graph health and each node's status w.r.t. their topological role in this graph. As an example, it can be easily used as a bottleneck detection mechanism. To the best of our knowledge it is the first distributed algorithm to provide such a complete information on topological health of a graph.

The convergence time reduction will be our next focus point ; directions include multiple "anomynous" random walks to decrease graph cover time [AF].Taking into account a dynamic environnment is another key point : the algorithm should be reactive enough so that the observations of the conductivity are consistent with the current network state. Finally, efficient repair mechanisms are of particular importance.

Appendix

Références

[Adl] Stephen Adler. The slashdot effect, an analysis of three internet publications. http ://hup.hu/old/stuff/slashdotted/SlashDotEffect.html.

[AF] David Aldous and James Allen Fill. Reversible markov chains and random walks on graphs.

[CPK07] Julian Candia, Paul E. Parris, and V. M. Kenkre. Transport properties of random walks on scale-free/regular-lattice hybrid networks. *J.STAT.PHYS.*, 129 :323, 2007.

[DS00] Peter G. Doyle and Laurie J. Snell. Random walks and electric networks, Jan 2000.

[ER59] P. Erdos and A. Renyi. On random graphs. In *Publicationes Mathematicae*, pages 6 : 290–297, 1959.

[Fei95] Feige. A tight upper bound on the cover time for random walks on graphs. *RSA : Random Structures & Algorithms*, 6, 1995.

[Has70] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1) :97–109, 1970.

[JFDG07] Mahdi Jafarisiavoshani, Christina Fragouli, Suhas Diggavi, and Christos Gkantsidis. Bottleneck discovery and overlay management in network coded peer-to-peer systems. In *INM '07 : Proceedings of the 2007 SIGCOMM workshop on Internet network management*, pages 293–298, New York, NY, USA, 2007. ACM.

[LCC+02] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *ICS '02 : Proceedings of the 16th international conference on Supercomputing*, pages 84–95, New York, NY, USA, 2002. ACM.

[MAS06] Damon Mosk-Aoyama and Devavrat Shah. Computing separable functions via gossip. In *PODC '06 : Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 113–122, New York, NY, USA, 2006. ACM.

[MMKG06] Laurent Massoulié, Erwan Le Merrer, Anne-Marie Kermarrec, and Ayalvadi Ganesh. Peer counting and sampling in overlay networks : random walk methods. In *PODC '06 : Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 123–132, New York, NY, USA, 2006. ACM.

[Moh97] B. Mohar. Some applications of laplace eigenvalues of graphs, 1997.

[MPHD06] Alan Mislove, Ansley Post, Andreas Haeberlen, and Peter Druschel. Experiences in building and operating epost, a reliable peer-to-peer application. In *EuroSys '06 : Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems 2006*, pages 147–159, New York, NY, USA, 2006. ACM.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge University Press, New York, NY, USA, 1995.

[New02] M. E. J. Newman. Random graphs as models of networks. Working Papers 02-02-005, Santa Fe Institute, February 2002.

[PC03] Jeffrey D. Picka and Karthik Chermakani. Random-walk-based estimates of transport properties in small specimens of composite materials. *Phys. Rev. E*, 67(4) :041104, Apr 2003.

[Sin92] Sinclair. Improved bounds for mixing rates of markov chains and multicommodity flow. In *LATIN : Latin American Symposium on Theoretical Informatics*, 1992.

[SLS06] Min Sheng, Jiandong Li, and Yan Shi. Critical nodes detection in mobile ad hoc network. In *AINA '06 : Proceedings of the 20th International Conference on Advanced Information Networking and Applications - Volume 2 (AINA'06)*, pages 336–340, Washington, DC, USA, 2006. IEEE Computer Society.

[SRD+06] Daniel Stutzbach, Reza Rejaie, Nick Duffield, Subhabrata Sen, and Walter Willinger. On unbiased sampling for unstructured peer-to-peer networks. In *IMC '06 : Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 27–40, New York, NY, USA, 2006. ACM.

[SW97] Mechthild Stoer and Frank Wagner. A simple min-cut algorithm. *J. ACM*, 44(4) :585–591, 1997.

[vid] http ://www.irisa.fr/asap/members/gtredan/output.avi/.

A Figures

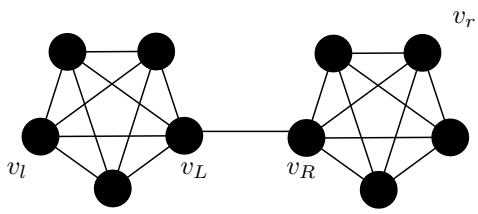

FIG. 1 – A barbell graph

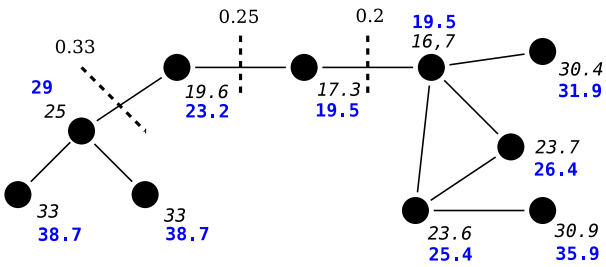

FIG. 2 – Conductance and standard deviation of return times on a micro network with n=10 nodes

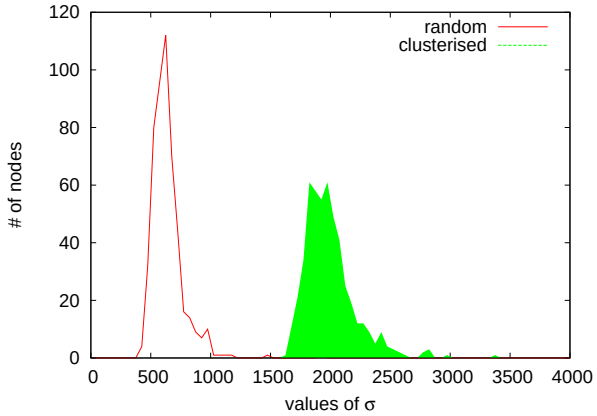

FIG. 3 – Distribution of standard deviations, on both random and clusterized graphs with n=1000

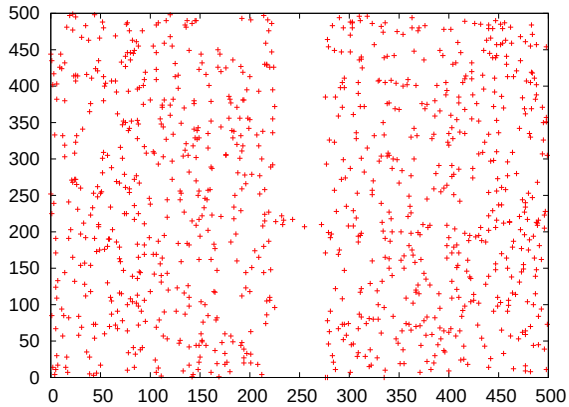

FIG. 4 – Repartition of 1000 nodes on a bottlenecked 2-D topology

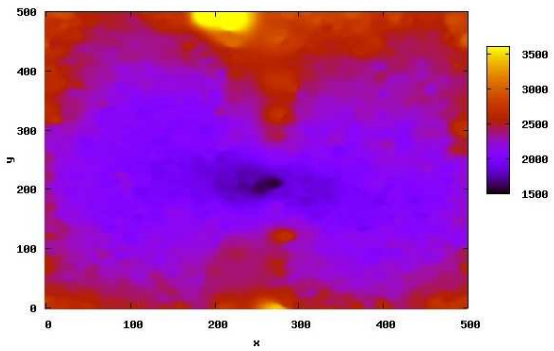

FIG. 5 – Resulting distribution of standard deviations on network nodes, after 3×10^6 probe's steps

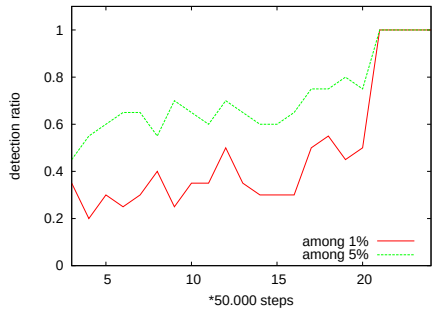

FIG. 6 – Evolution of the position of bridges nodes among the lowest standard deviations of all network nodes

B Proofs

B.1 Proof of Theorem 1

By definition of $f_j^{(n)}(i)$ we have, for $n = 1$, $f_j^{(1)}(i) = P(i,j)$. For $n \geq 2$, we have

$$\begin{aligned}
f_j^{(n)}(i) &= \mathbb{P}\{\tau(j) = n \mid X_0 = i\} \\
&= \mathbb{P}\{X_n = j, X_k \neq j, 1 \leq k \leq n-1 \mid X_0 = i\} \\
&= \sum_{\ell \in S-\{j\}} \mathbb{P}\{X_n = j, X_k \neq j, 2 \leq k \leq n-1, X_1 = \ell \mid X_0 = i\} \\
&= \sum_{\ell \in S-\{j\}} \mathbb{P}\{X_1 = \ell \mid X_0 = i\}\mathbb{P}\{X_n = j, X_k \neq j, 2 \leq l \leq n-1 \mid X_1 = \ell, X_0 = i\} \\
&= \sum_{\ell \in S-\{j\}} P(i,\ell)\mathbb{P}\{X_n = j, X_k \neq j, 2 \leq k \leq n-1 \mid X_1 = \ell\} \\
&= \sum_{\ell \in S-\{j\}} P(i,\ell)\mathbb{P}\{X_{n-1} = j, X_k \neq j, 1 \leq k \leq n-2 \mid X_0 = \ell\} \\
&= \sum_{\ell \in S-\{j\}} P(i,\ell) f_j^{(n-1)}(\ell),
\end{aligned}$$

where the last but one and the antepenultimate equalities come respectively from the Markov property and the homogeneity of the Markov chain X. ■

B.2 Proof of Corollary 2

Using Relation (2), we obtain

$$\begin{aligned}
M_j &= \sum_{n=1}^{\infty} n f_j^{(n)} \\
&= P_j + \sum_{n=2}^{\infty} n Q_j f_j^{(n-1)} \\
&= P_j + Q_j \left(\sum_{n=2}^{\infty} (n-1) f_j^{(n-1)} + \sum_{n=2}^{\infty} f_j^{(n-1)} \right) \\
&= P_j + Q_j \left(\sum_{n=1}^{\infty} n f_j^{(n)} + \sum_{n=1}^{\infty} f_j^{(n)} \right) \\
&= P_j + Q_j \left(M_j + \mathbb{1} \right),
\end{aligned}$$

and, since $P_j + Q_j \mathbb{1} = \mathbb{1}$, we get

$$M_j = Q_j M_j + \mathbb{1}.$$

Matrix Q_j is the submatrix of the transition probability matrix of an absorbing Markov chain with $|S|$ transient states and one absorbing state, thus the matrix $I - Q_j$ is invertible. This leads to

$$M_j = (I - Q_j)^{-1} \mathbb{1}.$$

■

B.3 Proof of Corollary 3

Using Relation (2), we obtain

$$\begin{aligned}
H_j &= \sum_{n=1}^{\infty} n^2 f_j^{(n)} \\
&= P_j + \sum_{n=2}^{\infty} n^2 Q_j f_j^{(n-1)} \\
&= P_j + Q_j \left(\sum_{n=2}^{\infty} (n-1)^2 f_j^{(n-1)} + 2 \sum_{n=2}^{\infty} (n-1) f_j^{(n-1)} + \sum_{n=2}^{\infty} f_j^{(n-1)} \right) \\
&= P_j + Q_j \left(\sum_{n=1}^{\infty} n^2 f_j^{(n)} + 2 \sum_{n=1}^{\infty} n f_j^{(n)} + \sum_{n=1}^{\infty} n f_j^{(n)} \right) \\
&= P_j + Q_j \left(H_j + 2M_j + \mathbb{1} \right) \\
&= Q_j H_j + 2Q_j M_j + \mathbb{1} \\
&= Q_j H_j + Q_j M_j + M_j,
\end{aligned}$$

since, from Corollary 2, we have $Q_j M_j + \mathbb{1} = M_j$. This leads to

$$H_j = (I - Q_j)^{-1}(I + Q_j) M_j.$$

■

Table des matières

1 Motivations **3**

2 System model **4**

3 Related work **4**

4 Perpetual random walks **5**
4.1 Intuition 5
4.1.1 The high clustering intuition 5
4.1.2 The electric lattice intuition 6
4.2 The permanent probe algorithm 6
4.2.1 Unbiased random walk 6
4.2.2 Standard deviation of return times 7

5 Return times in Markov chains **7**

6 Evaluation **10**
6.1 Execution on a micro example 10
6.2 Simulations on pathologic graphs 10

7 Discussion **11**
7.1 Algorithm convergence time 12
7.2 Distributed detection 12
7.3 Conclusion and future work 12

A Figures **III**

B Proofs **V**
B.1 Proof of Theorem 1 V
B.2 Proof of Corollary 2 V
B.3 Proof of Corollary 3 VI

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399