Fully Decentralized Estimation of Some Global Properties of a Network

Antonio Carzaniga
University of Lugano
Lugano, Switzerland
Email: antonio.carzaniga@usi.ch

Cyrus Hall
University of Lugano
Lugano, Switzerland
Email: hallcp@acm.org

Michele Papalini
University of Lugano
Lugano, Switzerland
Email: michele.papalini@usi.ch

***Abstract*—It is often beneficial to architect networks and overlays as fully decentralized systems, in the sense that any computation (e.g., routing or search) would only use local information, and no single node would have a complete view or control over the whole network. Yet sometimes it also important to compute global properties of the network. In this paper we propose a fully decentralized algorithm to compute some global properties that can be derived from the *spectrum* of the network. More specifically, we compute the most significant eigenvalues of a descriptive matrix closely related to the adjacency matrix of the network graph. Such spectral properties can then lead to, for example, the "mixing time" of a network, which can be used to parametrize random walks and related search algorithms typical of peer-to-peer networks. Our key insight is to view the network as a linear dynamic system whose impulse response can be computed efficiently and locally by each node. We then use this impulse response to identify the spectral properties of the network. This algorithm is completely decentralized and requires only minimal local state and local communication. We show experimentally that the algorithm works well on different kinds of networks and in the presence of network instability.**

I. Introduction

A peer-to-peer system is essentially a network of applications used to communicate and share resources. The nodes in this network, called "peers," are typically interconnected through point-to-point overlay links and typically maintain only a local view of the network consisting of a little more than a list of adjacent peers. Therefore, a peer-to-peer system is a good example of a decentralized network. Their decentralized nature makes peer-to-peer systems ideal for emergent and naturally decentralized environments (e.g., infrastructureless or end-user networks) and also makes them particularly resilient to failures. On the other hand, for some core network functions (e.g., maintenance and search) it is still desirable that peers be able to measure certain global properties of the network.

In this paper we propose a method to measure some global properties of a network such as a peer-to-peer system in a completely decentralized and efficient manner, where decentralized and efficient means that the memory required at each node and the traffic on each link are asymptotically sublinear in the size of the network and practically very small.

We focus on properties of a networks that can be derived from the *spectrum* of the network. By spectrum of the network we mean the eigenvalues of a chosen descriptive matrix closely related to the adjacency matrix of the network graph. For example, one useful property that can be derived from the spectrum of the network is the network's *mixing time*. The mixing time is the number of hops after which a random walk becomes independent of its starting point, and is an important parameter of many search and gossip algorithms. The mixing time can be derived from the second-largest eigenvalue modulus of the transition probability matrix, which is simply a weighted adjacency matrix of the network graph. Similarly, other useful properties of the network can be derived from the spectrum of the same and other related matrices. Kempe and McSherry discuss some such properties in detail [1].

Our intuition is that each node in the network can, with a simple, efficient, and fully decentralized algorithm, compute the impulse response of a particular linear dynamic system whose state-transformation matrix corresponds to the chosen descriptive matrix derived from the network graph. In other words, a node can not see or use the whole network graph (efficiently) but can compute the impulse response of a dynamic system whose behavior is defined by the structure of that graph. This impulse response can then be used to compute the desired salient spectral properties of the network graph by applying well-established identification and realization techniques developed within the theory of dynamic systems.

In general, a complete and exact identification would require a very large number of steps of the impulse response (at least $2n - 1$ for a network of n nodes). However, as it turns out, a much shorter, truncated impulse response is sufficient to reveal useful information in low-diameter networks, which include essentially every network of practical use. In this paper we demonstrate experimentally that this is the case for some important classes of networks and for significant network sizes. Furthermore, we show that our technique is, to some extent, resilient to network instability (or "churn").

We now discuss in Section II how this work relates to research in the computation of global network properties as well as in distributed spectral analysis. Then, in Section III we declare the basic models we rely on, including the model of the network. Based on these models, in Section IV we formally describe the problem we are solving and detail the algorithm we propose to solve it. In Section V we present the results of the algorithm's empirical evaluation, and in Section VI we conclude with a discussion of open problems.

978-1-4673-0775-8/12/$31.00 ©2012 IEEE

II. Related Research

The core idea of this paper is to apply system identification and realization techniques to the estimation of global properties of networks such as peer-to-peer systems. In this section we first discuss other ideas on how to estimate global properties in a decentralized manner. Then we review specific methods that are relevant to the computation of the spectrum of a network. Our review of related research is set within the context of peer-to-peer systems, even though most of the ideas and techniques developed in this paper are more generally applicable to other, conceptually similar networks (e.g., sensor networks, computational clusters) and to communication networks in general.

A. Monitoring and Aggregate Measurements

One way to measure global properties of a peer-to-peer system is to sample local properties and then to infer global projections. Another way is to compute global aggregates of data distributed over the network using specific aggregation algorithms. Sometimes, these techniques use the same basic notions or the same building blocks, as in the case of random walks and gossip algorithms.

We start with aggregation algorithms. One of the foundational algorithms of this kind was proposed by Kempe et al. and is called *Push-Sum* [2]. Push-Sum is a probabilistic algorithm for the computation of various aggregates, such as the sum and average of node data. Each node v starts with a value x_v and builds, through the running of the algorithm, an estimate of some global value, such as the sum $\sum_v x_v$ or the average $1/n \sum_v x_v$ of x_v over the network (n is the size of the network). Push-Sum is simple and, given a desired error bound, converges in $O(\log n)$ rounds. Push-Sum has also been improved and extended [3]. Mosk-Aoyama and Shah developed an algorithm that also computes sums in a completely distributed manner [4]. Their algorithm extends to the more general class of *separable functions* over node data and to any fixed topology, whereas Push-Sum requires complete connectivity. Separable functions can be expressed as the sum of individual functions, and therefore include summation and averaging.

Aggregation algorithms are undoubtedly very useful. However, like other techniques described in this section, they are limited to simple functions of node data (e.g., sum, average) and do not seem to be directly applicable to properties of the network that are more intrinsically dependent on its connectivity and structure (e.g., spectral properties). Aggregation algorithms can, however, serve as basic building blocks for the computation of such properties (see Section II-B).

Within the rubric of sampling, it is useful to distinguish two broad categories of techniques that operate on peer-to-peer systems. The first category includes techniques developed for specific systems or topologies. Conversely, the second category includes general-purpose techniques, which we further classify in two main subcategories: those based on random walks, and those based on gossip protocols.

King and Saia present a method to select a node uniformly at random from any distributed hash table (DHT) that supports the common ID-based lookup function where $get(x)$ returns the closest node to ID x [5]. Similarly, studies of churn dynamics in DHTs often assume that nodes can be sampled by looking up randomly selected values in the address space [6]. However, many DHTs show a preference toward storing long-lived nodes in routing tables in order to increase reliability, which biases the selection toward such nodes. Further, natural differences in the distribution of IDs biases toward nodes that are responsible for larger portions of the ID-space. In general, any technique which samples nodes in a manner correlated with session length will lead to biased results [7], [8]. Beyond their statistical properties, these techniques are also limited in that they are only applicable to DHTs.

Moving away from system-specific techniques, Massoulié et al. develop two generic techniques to solve the peer-counting problem, namely the estimation of the size n of a peer-to-peer network [9]. The first technique, called *Random Tour*, which also extends to summations, exploits a property of random walks. More specifically, the sampling is done through a random *tour*, which can estimate the size of the network essentially because the *hitting time* (i.e., the expected number of hops before a random walk goes back to the originating node) relates to the global visitation probability of the originator node and can be computed locally. The second technique, also based on sampling and called *Sample and Collide*, is an extension of a technique by Bawa et al. [10] based on the birthday paradox. In essence, the algorithm measures the expected number of uniform samples taken before a collision is recorded (i.e., before the same node is sampled twice). Then, by the birthday paradox, the total size n is computed as the square of the expected collision sample size.

Beyond the use of specific properties of random sampling and random walks, such as the birthday paradox and hitting time, the most common approach to estimate a globally distributed property is to measure it over a statistically significant portion of the network. And a common way to do that is once again to use random walks, where the property is sampled at a node selected at the end of a sufficiently long random walk. The cost and precision of this sampling technique depend on the mixing time of the network, which is itself a property that can not be computed through simple aggregation, and that leads us back to the main objective of this paper.

B. Distributed Spectral Computations

The techniques we review now are particularly relevant to the work presented in this paper because they are intended to compute structural properties of the network, and more specifically properties of the spectrum of the network graph.

Kempe and McSherry propose to compute the top-k eigenvectors of a matrix distributed across a network using a fully distributed version of the traditional orthogonal iteration algorithm [1]. Instead of using QR decomposition to orthonormalize in each round, Kempe and McSherry use Push-Sum [2] to estimate a matrix K related to the system, which

is then factored by each node to find the orthonormalization matrix. The algorithm halts by detecting convergence in a distributed fashion. Since it computes the *eigenvectors* of a distributed matrix, the algorithm of Kempe and McSherry is clearly very powerful. By contrast, the method we propose in this paper is intended to provide more high-level information, since it only computes the top-k *eigenvalues* (see Section IV-A below for more details). However, we should note that the algorithm of Kempe and McSherry is also slower, roughly by a factor of at least $\log^2 n$, and significantly more expensive in terms of traffic. Also, notice that the impulse-response method we propose here does not inherently exclude the computation of the eigenvectors. Rather, it is the computation of the eigenvectors of the *full* system, with its $\Omega(n)$ memory requirement for each node, that does not match our basic efficiency requirements.

EigenSpeed and EigenTrust both use distributed power iteration to find the first eigenvector of a matrix [11], [12]. Both EigenSpeed and EigenTrust approach the problem of malicious nodes, which we do not address in this paper. While the power method can be used to find more than just the first eigenvector, each additional eigenvector must be run sequentially, making the process slow and expensive. Also, this process is not evaluated in the presence of churn.

III. Preliminaries

We model a network as a directed graph $G = (V, E)$ over n vertexes, where each vertex represents a node and each edge represents a one-way link between two nodes. (A duplex link is represented by two edges.) We use $deg(v)$ to denote the out-degree of node v. We make two assumptions regarding the network graph. First, we assume that the network is ergodic. Formally, this means that G is strongly connected and aperiodic. That is, the greatest common divisor of the lengths of its cycles is 1. In practice, this means that, starting from any node, a sufficiently long walk can reach every other node and it can do so at every (sufficiently long) length. Second, we focus on networks with low diameter $\Delta \ll n$ and, less crucially, low (maximum) degree d. These assumptions are quite reasonable in practice. For example, practically all peer-to-peer networks have diameter $\Delta = O(\log n)$ and many of them have average degree $d = O(\log n)$.

The estimation technique we propose uses classic methods from system identification and realization. Therefore, we review the basic model of a dynamic system that we use. We consider a linear time-invariant dynamic system defined by the following state-space equations:

$$x(t+1) = Ax(t) + Bu(t) \quad (1)$$

$$y(t) = Cx(t) \quad (2)$$

More specifically, this is a discrete-time, linear, time-invariant, single-input-single-output, deterministic system in which $x(t) \in \mathbb{R}^n$ is a vector representing the state of the system at time t, $u(t) \in \mathbb{R}$ is a scalar representing the input at time t, and $y(t) \in \mathbb{R}$ is the output; $A \in \mathbb{R}^{n \times n}$ is the state-transformation matrix, $B \in \mathbb{R}^n$ is a vector that maps the input value into the state of the system, and $C \in \mathbb{R}^n$ is a vector that maps the state into the output. The size n of the state vector x is also called the *order* of the system.

We denote the impulse response of the system with $h(t)$ for $t = 1, 2, \ldots$ starting from the quiescent state $x(0) = 0$. Thus, $h(t)$ is the output of the system when the input is the unit impulse ($u(0) = 1$ and $u(t) = 0$ for $t > 0$) and can be written as $h(t) = CA^{t-1}B$ (for $t = 1, 2, \ldots$).

IV. Spectral Estimation

We propose an algorithm to compute global network properties using the spectral properties of the adjacency matrix of the network graph. In this paper we focus our evaluation on a property called the mixing time of the network, which is related to the second-largest eigenvalue modulus of a matrix closely related to the adjacency matrix of the network. However, as we will argue below, the algorithm can also be applied to other matrices related to the adjacency matrix, and therefore to other properties of the network studied in spectral graph theory.

A. Problem Statement

Given a network graph G, we say that a matrix $A = (a_{uv})$ is related to the adjacency matrix of G (or simply related to G) if $a_{uv} \neq 0$ only if there is an edge (v, u) in G or if $u = v$. Notice that, contrary to other models, we associate the position (u, v) in the adjacency matrix to the (v, u) edge in G. As should become clear later, we do that for consistency with the standard model of equations (1) and (2). We are interested in matrices related to G that are induced by local information held by nodes in the network. Thus, intuitively, each column $(a_{\cdot v})$ of A is associated with node v and represents the outgoing edges of v (correspondingly, each row $(a_{u\cdot})$ represents the incoming edges of u).

Given a matrix A related to G and stored column-wise by the nodes of G—meaning that each node v stores the v-column of A—we propose an algorithm to compute the dominant spectral components of A, namely the largest eigenvalues of A (largest in modulus).

As an example, consider a random walk on G. Such a walk can be modeled as a Markov chain with transition matrix $P \in [0, 1]^{n \times n} = (p_{uv})$, where p_{uv} is the probability to go from node v to the node u in a step of the random walk. In an unbiased random walk, P is related to the adjacency matrix of G as follows: $p_{uv} = 1/deg(v)$ for all edges (v, u) in G, and $p_{uv} = 0$ otherwise. In general, each column $p_{\cdot v}$ in P can define any probability distribution over the outgoing edges of node v. In either case, each node v decides how to bias the probability of walking from v to all its neighbors, and therefore effectively holds the column $p_{\cdot v}$ in P. This is how random walks are implemented in many peer-to-peer systems. The algorithm we propose would compute the first two eigenvalues of P and then return the second one, which determines the mixing time of the network and therefore the minimal length of a random walk necessary to achieve the desired approximation of the asymptotic visitation probability over G. (The algorithm would

compute the first two eigenvalues, but in this case the first is not used because it is known to be always equal to 1, since P is a stochastic matrix.)

B. Estimation Algorithm

The algorithm could identify the full spectrum of A exactly. However, in practice it would be used to produce an estimate of the dominant eigenvalues. Therefore, we refer to it as an estimation algorithm. The algorithm consists of three phases: a first phase consisting of k rounds, denoted with $t = 1, 2, \ldots, k$, in which each node in the network exchanges a message with each of its neighbors; a second phase in which nodes perform a local calculation that yields the estimated eigenvalues; and a third optional phase in which nodes may engage in a one-hop gossip to refine their estimate based on those of their neighbors.

We now describe the algorithm assuming that the execution of each round is synchronous, meaning that all nodes execute round t at the same time, and finish that execution within a bounded interval. In particular, all nodes start from the first iteration ($t = 1$) at the same time. In Section IV-C we discuss how the algorithm can be implemented without global synchrony and coordination.

Each node v maintains a scalar variable x_v. Node v also holds the v-column $(a_{\cdot v})$ of the target matrix A. Notice that this is a sparse column with only a few non-zero values corresponding to the out-edges of v.

Algorithm 1 estimation algorithm executing at node v

```
1:  x_v ← choose a value from {0, 1} uniformly
2:  h_v(1) ← x_v
3:  for t ← 2 ... k do
4:      for u ∈ out-neighbors(v) do
5:          send value x_v a_uv to u
6:      end for
7:      collect all values w sent by in-neighbors
8:      x_v ← Σ w
9:      h_v(t) ← x_v
10: end for
11: Â_v = Kung's realization with h_v(1), ..., h_v(k)
12: compute the dominant eigenvalues of Â_v
13: exchange the eigenvalues with neighbors
14: collect estimates from neighbors
15: adjust estimates to the median of the collected estimates
```

With this local state and local input, the first phase of the algorithm proceeds as follows (see Algorithm 1). Each node v initializes x_v to 0 or 1 (line 1); then at every round, each node v sends the value $x_v a_{uv}$ to each node u in its out-neighborhood (line 5) and then updates its value x_v with the sum of the values received from its in-neighbors (line 8). Each node v also records the value of x_v for each round as $h_v(t)$ for $t = 1, 2, \ldots, k$.

Notice that, denoting with $x(t)$ the column vector of all the node-local values x_v at round t, each round of the algorithm amounts to a distributed computation of the linear transformation $x(t+1) = Ax(t)$, where the initial value $x(1)$ is the vector of zeros and ones each chosen randomly by a node. In other words, for each node v, the sequence $h_v(1), h_v(2), \ldots h_v(k)$ computed by the algorithm corresponds to the impulse response of the following linear time-invariant dynamic system:

$$x(t+1) = Ax(t) + Bu(t) \quad (3)$$

$$y(t) = C_v x(t) \quad (4)$$

where $B = x(1)$ and C_v is a row vector of all zeroes except for a 1 in position v, and therefore simply extracts x_v from x.

Notice also that, although the algorithm requires all nodes to participate in the computation of the impulse responses, a node that is not interested in computing the spectral properties of A does not have to store the impulse response (lines 2 and 9) and therefore may terminate the execution after the first phase (line 10). Thus, after the first phase, each node v that is indeed interested in computing the spectral properties of A holds the impulse response $h_v(t)$ of a particular node-specific dynamic system whose state-transition matrix is the target (unknown) matrix A.

In the second phase, node v uses $h_v(t)$ to compute a realization of a surrogate system with the same dynamic behavior as the system of equations (3) and (4). This computation uses Kung's classic realization algorithm [13], [14]. Kung's algorithm produces an approximation of a linear system in the state-space, starting from a truncated impulse response of that system. In particular, given $k = 2l - 1$ values of the impulse response $h_v(1), h_v(2), \ldots, h_v(2l-1)$, Kung's algorithm computes a triple $\hat{A}_v, \hat{B}_v, \hat{C}_v$ of matrices that define a system that approximates the system defined by equations (3) and (4) but possibly of lower order $\hat{n} \leq n \leq l$. In our algorithm we ignore $\hat{B}_v$ and $\hat{C}_v$ and use only $\hat{A}_v$ to compute its spectrum using a standard numeric linear-algebra method.

Notice that Kung's algorithm requires an impulse response of at least $k = 2n - 1$ steps to realize a system of order n. This means that a full and exact estimation would require an impulse response twice the size of the network. However, we propose to use much shorter impulse responses. In particular, we propose to use k on the order of the *diameter* of the network, as opposed to its size n. In other words, we propose a non-standard use of Kung's algorithm.

This aspect of the algorithm is a bit delicate. The essence of the problem is that we want to identify the dominant spectral properties of a high-dimensional system (i.e., a large network) using only a few steps of its impulse response. Although it is known that a short impulse response may not lead to a valid approximation in all cases, such method seems justified within our chosen application domain. As we argued in Section III, networks are by nature ergodic and have low-diameter. Thus, our intuition is that even a short prefix of the impulse response—one that would have to be significantly larger than the diameter of the network, but still much smaller than the size of the network—would contain enough information about the entire network to allow for a

good approximate realization. This intuition is confirmed by the experimental results we present in Section V. However, establishing a more formal relation between the length k of the impulse response, the diameter of G, and the quality of the resulting realization remains an open problem.

As it turns out, the estimation is not uniformly good across the network, meaning that different nodes observe different systems that then lead to different surrogate realizations, which in turn lead to different spectral estimations with different levels of accuracy. The third and last phase of the algorithm is intended to make the estimate a bit more uniform across the network through a simple one-hop "gossip" exchange between nodes. In particular, each node exchanges its estimates of the dominant eigenvalues with its neighbors, and also collects the estimates of its neighbors. Then each node adjusts its estimates to the median of the estimates of its neighbors plus its own. (The median is applied to the moduli of the computed eigenvalues.)

C. Initiation and Coordination

The computation of the impulse response, which we described as a completely synchronous process, can also be initiated and carried out in a completely decentralized and asynchronous environment. Initiation and execution are quite simple if one can assume that all nodes have synchronized clocks. In that case, the initiation could be scheduled ahead of time (e.g., on a regular basis at fixed times) or it can be requested from a single node by broadcasting an absolute starting time through a controlled flood. The execution could then proceed with each round executed in agreed-upon time intervals.

The situation is only a bit more involved if one does not assume synchronized clocks. In this case we do not assume any agreed-upon schedule but we still assume that all nodes complete the execution of each round (a trivial computation) in a bounded amount of time. In this setting, the computation is initiated by a requesting node with a broadcast message transported through a controlled flood. This controlled flood is such that each node v sends one broadcast to its out-neighbors, and correspondingly receives a copy of the same from all nodes in its in-neighborhood. In fact, this initial flood can also carry the initial value $x_v(1)$ for all nodes v. Therefore, after a reasonable bootstrap interval, each node v has identified its in-neighborhood and can proceed with the computation at round $t = 2$. For all subsequent rounds $t > 2$, the computation at each node v can make progress in a completely decentralized and asynchronous manner with v sending out its round-t message as soon as v receives all round-$(t-1)$ messages from its in-neighbors.

Notice that the initial bootstrap interval can be reasonably short, since that must be proportional to the network diameter, which we assume to be small. Notice also that the purpose of the bootstrap interval is to establish in-neighborhoods, so if one can assume that in-neighborhoods are known, the computation may proceed immediately without the initial bootstrap interval.

V. Evaluation

In this section we present the results of the experiments we conducted to evaluate the spectral estimation algorithm with different kinds of graphs. We start by looking at ideal networks, where no faults or node disconnections can occur during the execution of the algorithm. We show that in this case our algorithm provides a good approximation of the desired global property using a short impulse response. We also show how the additional gossip round affects the results by reducing the variance in the estimation between the nodes. Then we look at the behavior of the algorithm in the presence of churn. In particular, we study what happens when one failure occurs during the algorithm execution and we study how the failure affects the estimates at different points in the computation of the impulse response.

A. Experiment Setup

In order to validate the estimation algorithm we test it with three different kinds of graphs:

- *Barabási-Albert*: This class of graphs are generated using a preferential-attachment model: a new node connects to high-degree nodes with higher probability [15]. We use a preferential factor of $1/2$.
- *Erdős-Rényi*: This is a random graph where an edge exists between two nodes with a given probability p [15]. In our case, we set $p = \log n/n$, so as to obtain $n \log n$ edges in expectation.
- *Chord*: This graph emulates the topology generated in a Chord peer-to-peer system [16].

All the graphs we use satisfy our initial assumptions, and in particular they are strongly connected and ergodic. All the graphs have $n = 10{,}000$ nodes. In our experiments we consider a matrix corresponding to the transition probability matrix of the Markov model of an unbiased random walk over G. Therefore, all the edges coming out of a node v have the same transition probability $1/deg(v)$.

For each graph we study two closely related global properties. The first is the *spectral gap* (or simply gap) of the graph. This measurement is given by the difference between the two eigenvalues with largest moduli, λ_1 and λ_2. In our case, since we consider a stochastic matrix (all columns sum to 1) which is known to always have a first eigenvalue of $\lambda_1 = 1$, the spectral gap is defined as $g = 1 - |\lambda_2|$. The second property is the *mixing time* τ, that is the necessary length of a random walk needed to reach the stationary distribution of visitation probabilities within a certain error ε. More specifically, given a graph G, the probability of terminating at node v after a random walk of length t starting from node v_0 depends on v, t, and v_0. However, for longer and longer walks, in the limit for $t \to \infty$, the probability to reach v depends only on v. This probability is called the stationary distribution of G. Now, given a desired error ε, the mixing time is the minimum number of hops after which a random walk approximates the stationary distribution with a maximum difference of ε.

Given a graph G and its transition probability P, a way to compute the mixing time is through repeated matrix multiplication, since if $x(t)$ is the vector representing visitation probabilities at t, the same vector after one hop in the random walk is $x(t+1) = Px(t)$ and from an initial starting point represented by $x(0)$, $x(t) = P^t x(0)$. However, this method is very slow. Instead, approximations and bounds to the mixing time can be computed using the spectrum of P [12], [17], exploiting the eigenvalue decomposition of P. In particular, we find an upper bound of the mixing time considering the error ε as the maximum possible contribution from the distribution associated with the second eigenvector of P. This is because all components on the second and successive eigenvectors vanish to zero exponentially in t, and the component on the second eigenvector is the one that vanishes at the slowest rate. We consider $max(q_2)$, the maximum value in the normalized eigenvector associated with λ_2. In the worst case, where the random walk starts at the node which gives the maximum contribution $max(q_2)$, the node contribution decreases with rate $|\lambda_2|^t$, where t is the number of hops taken. So we can approximate the mixing time τ with t. Therefore, the maximum error ratio with respect to the stationary distribution is $|\lambda_2|^t$, and if we demand that ratio be lower than ε, then the mixing time must be $\tau \approx t = \log_{|\lambda_2|} \varepsilon$.

Note that a small error in the estimation of the spectral gap g may lead to a large error in the estimation of the mixing time τ, especially when $|\lambda_2|$ is close to 1 and therefore the spectral gap is close to 0. Therefore, it makes sense to study the accuracy of the estimation of both the spectral gap and the mixing time.

B. Ideal Networks

In the first experiment we test the estimation error of the spectral gap for each type of network graph. The results are shown in Figure 1, where figures 1a, 1c, and 1e on the left column show the error before the gossip round, while figures 1b, 1d, and 1f on the right column show the error after the gossip round. Each data set is labeled as follows: *ba* is for Barabási-Albert, *er* for Erdős-Rényi, and *chord* for Chord graphs. We plot the length k of the impulse response on the x-axis and the percent error on the y-axis, defined as

$$\left| \frac{\hat{g} - g}{g} \right| \times 100$$

where $\hat{g}$ is the estimated spectral gap value and g is the actual gap. In all the plots we show the 10th, 50th (median), and 90th percentile of the estimations of all the nodes in the network. More specifically, we repeat each experiment with 10 different graphs of the same type, and then we compute, for each length of k, the 10th, 50th, and 90th percentile of the estimations performed by all the nodes in all the graphs (100,000 nodes in total). The small inner plots (top-right corner) zooms into the results obtained for k between 60 and 120 (again, we have k on the x-axis and the percent error on the y-axis).

Erdős-Rényi graphs (figures 1a and 1b) show a fast convergence while Chord graphs (figures 1e and 1f) are slower to converge. In all cases, an impulse response of length $k \geq 60$ yields an acceptable estimation error for all the different graphs. However, regardless of the rate of convergence, the estimation incurs a fixed error that does not seem to depend on k, but that differs depending on the type of graph. For example, the estimation converges after $k \simeq 20$ for Erdős-Rényi graph (figures 1a and 1b), but converges to a value that is small but still positive. This is not a surprising behavior, since the estimated system has a much lower order than the real system, although as of yet, we do not completely understand the nature of this fixed error.

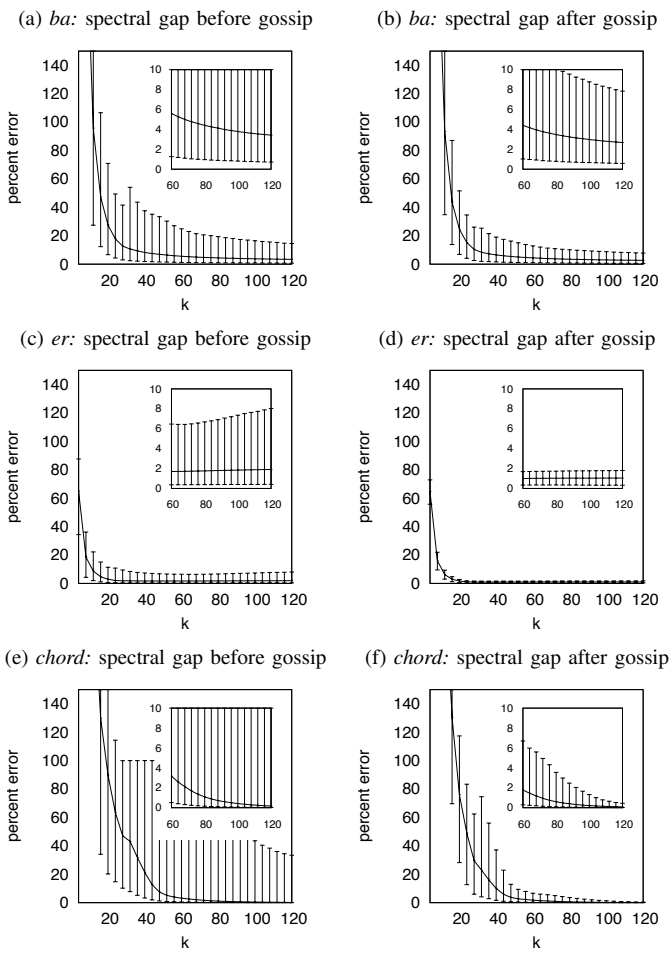

Fig. 1. Spectral gap estimation error

As for the one-hop gossip, a comparison between figures on the left column and figures on the right column shows that the one-hop gossip exchange is quite effective in reducing variability, especially in the graphs that seem to converge at the slowest rate (Chord, figures 1e and 1f).

In Figure 2 we show the same type of results, for the same experiments, but focusing on the computation of the mixing time instead of the spectral gap. Again, the figures on the left (2a, 2c, and 2e) show the results before the gossip reconciliation round, while the figures on the right (2b, 2d, and 2f) show the results after the gossip reconciliation round. We plot again the impulse response length k on the x-axis and

the percent error on the y-axis, defined in the same way as for the spectral gap.

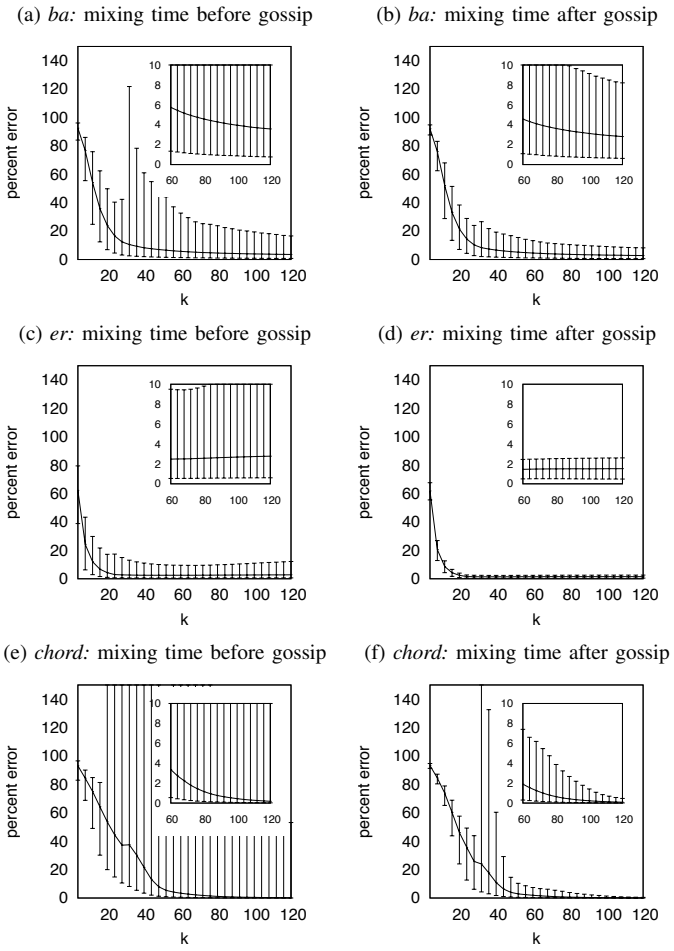

Fig. 2. Mixing time estimation error

We can see that, for short impulse responses, the median error is much smaller in the estimation of the mixing time than the estimation of the spectral gap. However, in the case of Chord (figures 2e and 2f) we observe a high variability highlighted by the difference between the 10th and 90th percentiles. This means that a few nodes compute bad estimates of the mixing time. Fortunately, as shown in Figure 2f, the gossip round is quite effective in mitigating this imprecision, and again after an impulse response of length $k \simeq 60$ we can obtain a good approximation for all the graphs.

We should also highlight a strange behavior in the case of the Chord graphs for impulse-response lengths around $k = 30$. Here we observe a significant peak of variability (i.e., inaccuracy) in the estimation. Probably, this depends on the fact that Chord graphs do not mix very well, which means that $|\lambda_2|$ is very close to 1, so even a small error in the estimation of λ_2 may result in a substantial error in the estimation of the mixing time. Nevertheless, this error disappears and the estimation converges after $k \geq 60$.

C. Network Instability (or Churn)

The previous analysis shows the accuracy of the estimation in an ideal network in which nodes and links are reliable and stable. However this is not always a realistic scenario. Even in the absence of failures, peer-to-peer networks are affected by churn, that is, nodes leaving or joining the network, thereby changing the adjacency matrix of the network that the algorithm intends to identify. In order to test whether our algorithm is resilient to these changes, we induce a failure corresponding to one node leaving the network at a certain time f_t during the computation of the impulse response. We then analyze the effect of the failure on the estimation of the spectral gap and mixing time. Notice that in this case, the estimation is performed at time $k > f_t$, by the remaining nodes, using the complete impulse response, which is effectively computed across two network configurations. Then, in order to evaluate the estimation errors, we choose the actual gap and mixing time of the configuration at time k, that is, after the failure.

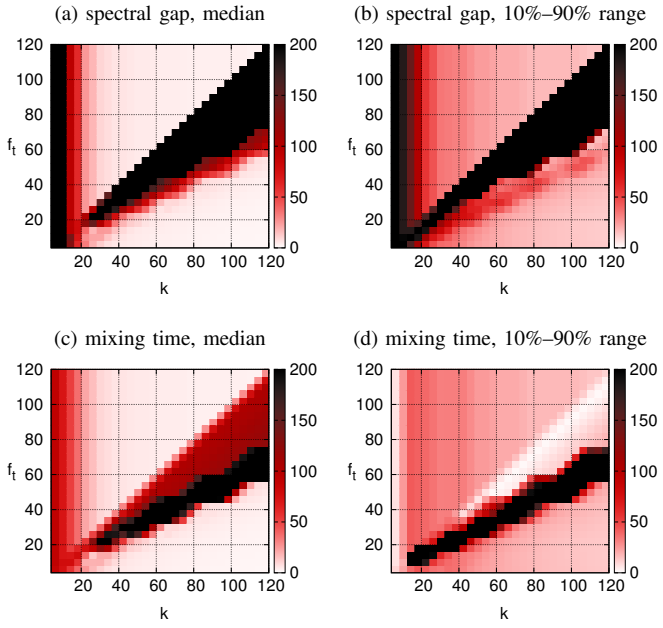

Fig. 3. Barabási-Albert graphs: estimation error with churn

In Figure 3 we see the effect of a failure in the case of Barabási-Albert graphs. We plot "heat maps" representing three-dimensional data. On the x-axis we plot the length k of the impulse response, and on the y-axis we indicate the time f_t at which the failure occurs. Thus, all the points above the diagonal correspond to failure-free estimations, while the points below the diagonal correspond to estimations based on an impulse response that extends over the failure. Each point in the map is color-coded with a gradient of colors representing the estimation error aggregated over all nodes across 10 different graphs of each type. Intuitively, the heat maps should be analyzed by fixing f_t and therefore by reading each chosen row left-to-right, with each point in the row

corresponding to the estimation error at that value of k.

Figure 3a represents the median percent-error of the spectral gap estimation computed over all nodes in 10 different graphs. The percent error is defined as before (e.g., as in Figure 1). Figure 3b shows the variability of the estimation computed as $|\hat{g}_{90\%} - \hat{g}_{10\%}|$ where $\hat{g}_{90\%}$ and $\hat{g}_{10\%}$ are the 90th and 10th percentile of the percent error of the spectral gap estimation. This way, we effectively measure the range that contains the central 80% of the estimates.

In figures 3c and 3d we plot the same values but for the mixing time. In order to test the worst situation possible, we induce the failure of node number 2 in the graph. Because of the way the graphs are constructed with the Barabási-Albert model, node 2, which is the second node added to the graph, is one of the nodes with the highest number of neighbors. So, a failure of this node affects the entire network in a significant way. We do not induce the failure of node 1 (which has even more neighbors) because that would typically partition the network. In all the other graphs (Erdős-Rényi and Chord) all nodes have the same number of neighbors with high probability, so we induce a failure in a randomly chosen node.

As shown in Figure 3a, the estimation error may be very high after a failure (more than 200% in same cases). However, with a sufficiently long impulse response, it is possible to recover and once again obtain a good approximation of the actual value. The range of variability (Figure 3b) is also quite low when the estimation reconverges. For the mixing time, we can draw essentially the same conclusions.

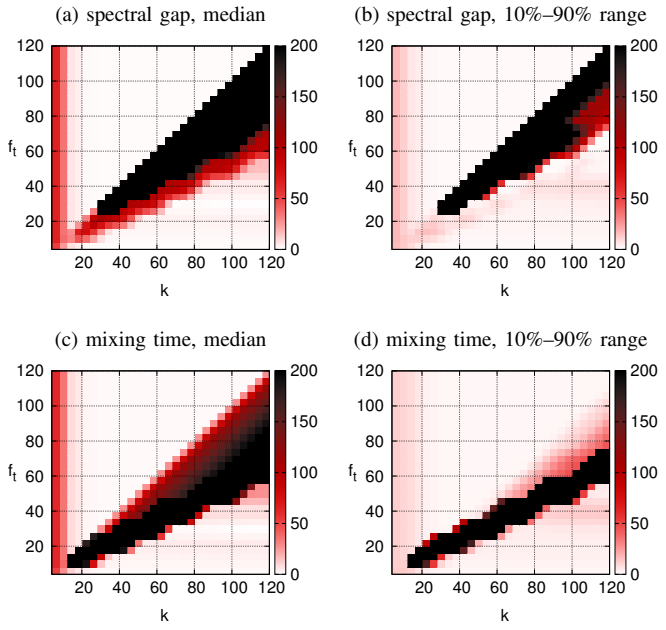

Fig. 4. Erdős-Rényi graphs: estimation error with churn

In figures 4 and 5 we present the same results for Erdős-Rényi and Chord graphs, respectively. Again the results show that the algorithm converges even in the presence of churn events, and in fact can also recover from an error with a reasonable length of the impulse response.

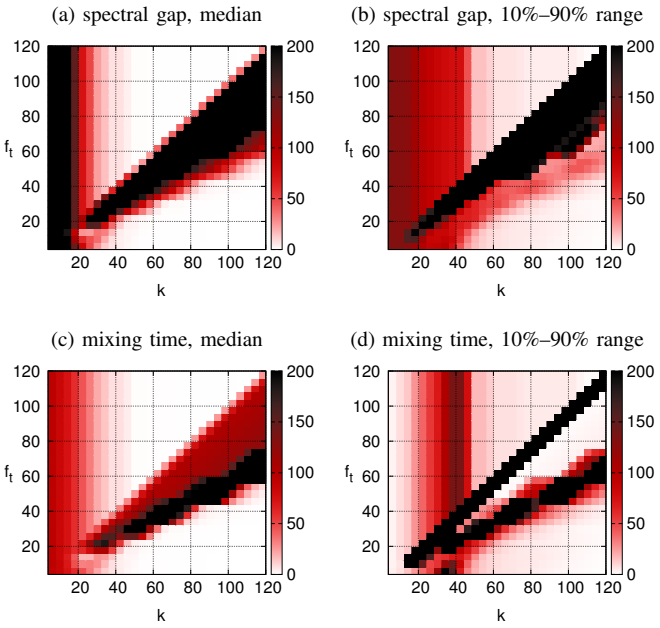

Fig. 5. Chord graphs: estimation error with churn

Erdős-Rényi graphs induce a behavior of the estimation algorithm similar to Barabási-Albert graphs, although the former type seem to give better results. In particular if the failure happens at the beginning of the computation ($k \leq 24$) the error in the gap estimation (figures 4a and 4b) remains relatively low and the algorithm converges again to the expected value in a few steps. In Erdős-Rényi graphs we also observe low variance, with most of the nodes computing the right value.

In the case of Chord graphs (Figure 5) we also obtain a good estimation with a sufficiently long tail of the impulse response after the failure. The main problem of this kind of graphs, however, is that the algorithm converges slowly to the best estimate, and not in a monotonic fashion, as evidenced in figures 1 and 2. This same behavior is highlighted also in Figure 5 in the presence of failures. Still, as we observed before, an impulse response of $k = 60$ is enough to obtain a good estimate, and this remains true even when we introduce a failure in the network. In fact, the number of steps needed to recover from a failure in the case of Chord graphs is comparable to the other two types of graphs.

VI. Conclusions

In this paper we presented an algorithm to estimate the spectral properties of a network. These properties can then be used to compute various useful global properties of the network. As an example, in this paper we focused on the mixing time, which is an important parameter of search algorithms based on random walks as well as other probabilistic distributed algorithms.

Our spectral estimation algorithm has a number of advantages. It is completely decentralized, as nodes maintain only local information (their adjacency list) and interact only locally; it requires a minimal amount of local state (a vector

of k real values or a single real value for nodes that are not interested in the estimation); it requires minimal and local communication (k rounds of very short messages sent on each link); it is simple, in the sense that it does not involve any complex interaction and it could be easily implemented so as to piggy-back the necessary communication on other traffic; and it also involves completely local computations that are efficient and can be carried out with common libraries. In short, the algorithm is simple and efficient.

The experimental evaluation we conducted shows that the algorithm is effective in estimating the dominant spectral characteristics of networks (specifically, the second eigenvalue) achieving good precision in 10,000-node networks with 60 rounds of communication, which in practice can be carried out in a few seconds. We then show, again experimentally, that the algorithm is also to a good extent resilient to network instability (specifically, individual node failures) although with varying results over different types of networks.

The research that lead us to develop this algorithm [18] also raises a few important questions that remain open. The most important ones are on the theoretical basis of the algorithm. In particular, the algorithm relies on the identification of a very high-dimensional system (the whole network) through an impulse response of much smaller dimensionality. Our conjecture is that this approach is generally valid whenever the diameter of the network is small. However, although our experimental analysis lends credibility to this conjecture, and in particular it shows a link between diameter and the necessary length of the impulse response, we do not have a proof that such a link is valid in general. Establishing such a link, or at least defining the theoretical bounds of applicability of our algorithm would be very useful and interesting, perhaps even beyond its applications in network research.

Exploring more practical developments, we also envision some ways to engineer the basic estimation algorithm so as to obtain even more robust and reliable results in the presence of churn. A potentially fruitful strategy, which we plan to explore in future research, is to compute and then use multiple, partially overlapping impulse responses. In algorithmic terms, this means running multiple instances of the estimation algorithm in a sort of pipeline, and then aggregating their individual outcomes appropriately. For example, using impulse responses of length $k = 120$, we may run 5 parallel instances of the estimation algorithm spaced in time at a distance of 20 iteration steps. Now, since failures seem to have negative effects only when the estimation is performed immediately after the failure, and since the parallel estimations are uniformly spaced in time, a single failure is likely to severely upset only a minority of the estimations. Therefore, a simple majority voting scheme might increase the precision of the estimate, making this parallel estimation almost completely oblivious to failures. Notice also that such parallel estimations would incur only a minimal cost for the additional local memory and processing, and practically no additional communication cost, since messages carrying parallel instances between two neighbors can be easily bundled.

ACKNOWLEDGMENTS

This work was supported in part by the Swiss National Science Foundation under grant number 200021-132565.

REFERENCES

[1] D. Kempe and F. McSherry, "A decentralized algorithm for spectral analysis," in *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, Jun. 2004.

[2] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)*, Oct. 2003.

[3] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: Design, analysis and applications," in *Proceedings of IEEE INFOCOM 2005*, Mar. 2005.

[4] D. Mosk-Aoyama and D. Shah, "Computing separable functions via gossip," in *PODC '06: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, Jul. 2006.

[5] V. King and J. Saia, "Choosing a random peer," in *Proceedings of the 23rd Symposium on Principles of Distributed Computing (PODC)*, 2004.

[6] R. Bhagwan, S. Savage, and G. Voelker, "Understanding availability," in *Proceedings of the 2nd International Workshop on Peer-To-Peer Systems (IPTPS)*, Jun. 2003.

[7] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger, "On unbiased sampling for unstructured peer-to-peer networks," in *Proceedings of the 6th Internet Measurement Conference (IMC)*, Oct. 2006.

[8] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *Proceedings of the 6th Internet Measurement Conference (IMC)*, Oct. 2006.

[9] L. Massoulié, E. Le Merrer, A.-M. Kermarrec, and A. 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*, Jul. 2006.

[10] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani, "Estimating aggregates on a peer-to-peer network," Stanford InfoLab, Technical Report 2003-24, Apr. 2003.

[11] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in P2P networks," in *Proceedings of the 12th International World Wide Web Conference (WWW)*, May 2003.

[12] R. Snader and N. Borisov, "EigenSpeed: Secure peer-to-peer bandwidth evaluation," in *Proceedings of the 8th International Workshop on Peer-to-Peer Systems (IPTPS)*, Apr. 2009.

[13] S. Kung, "A new identification and model reduction algorithm via singular value decomposition," in *Proceedings of the 12th Asilomar Conference on Circuits, Systems and Computers*, Nov. 1978.

[14] G. M. Pitstick, J. R. Cruz, and R. J. Mulholland, "Approximate realization algorithms for truncated impulse response data," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 34, no. 6, Dec. 1986.

[15] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of Modern Physics*, vol. 74, no. 1, 2002.

[16] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, 2001.

[17] S. Datta and H. Kargupta, "Uniform data sampling from a peer-to-peer network," in *27th IEEE International Conference on Distributed Computing Systems (ICDCS 2007)*, Jun. 2007.

[18] C. Hall and A. Carzaniga, "Uniform sampling for directed P2P networks," in *Euro-Par 2009*, Aug. 2009.