A Gossip-based Churn Estimator for Large Dynamic Networks

Cristiano Giuffrida, Stefano Ortolani

Department of Computer Science
Vrije Universiteit
Amsterdam, The Netherlands
{giuffrida,ortolani}@cs.vu.nl

Keywords: Gossiping, Churn, Aggregation.

Abstract

Gossip-based aggregation is an emerging paradigm to perform distributed computations and measurements in a large-scale setting. In this paper we explore the possibility of using gossip-based aggregation to estimate churn in arbitrarily large networks. To this end, we introduce a new model to compute local estimates and formally prove how aggregated values closely match the real churn with high accuracy independently of the network setting. Experimental results confirm the viability of our approach.

1 Introduction

Gossip-based protocols for large-scale communication are becoming increasingly popular due to their scalability and reliability properties. At the fundamentals of existing gossip-based protocol implementations lies a primary component called *peer sampling service* [10]. This service aims to provide an abstraction of a stream of randomly selected peers to the application layer, thus making a gossip-based protocol theoretically independent of network complexity and dynamics. In practice, most peer sampling service implementations target a specific network setting or are specifically optimized for particular scenarios (e.g. fault-tolerance, load-balancing). This clearly reduces the level of transparency at the application layer with respect to the conditions of the underlying network.

A step forward in this direction is provided by Jelasity et al. in [10]. Their work acknowledges the need to employ different policies at a peer sampling service level basing on the expected conditions of the network. They describe a generic peer sampling service implementation that can be tuned and optimized for different scenarios. However, their analysis develops under the assumption that responsibility is given at the application layer to tune the underlying peer sampling service properly; this is done a-priori, based on the expected conditions of the environment. This assumption, however, reduces the level of isolation of gossip-based applications from underlying services and may affect their portability to new deployment environments. Furthermore, a target environment may exhibit very different properties over time. Therefore, statically tuning a peer sampling service implementation to uniquely address specific properties may hardly be an effective solution.

The issues depicted above all come down to the way existing large-scale applications are typically structured. Most attempts to cope with network complexity and dynamics are targeted towards defining a robust application design that can proactively handle unpredictable situations. Gossip-based protocols have received growing attention due to the way they exploit and further develop this idea to an extreme. While this is clearly a crucial aspect in designing large-scale applications, we argue that it may not be the definitive solution.

Our vision contemplates a design in which large-scale applications can dynamically react to statically unpredictable conditions. To date, we believe this aspect has not received the required attention. For the applications to properly detect and cope with dynamics, we aim to introduce some level of network awareness in our system design. In this work, we explore the first step in this direction for gossip-based applications. More specifically, we concentrate our attention on a component that can provide realtime information about the conditions of the underlying network and can be seamlessly integrated into existing gossip-based solutions on top of a generic peer sampling service. We believe that the introduction of such com-

ponent may aid in the design of gossip-based solutions that can dynamically adapt to different settings and achieve a better separation of concerns between gossip-based services (e.g. peer sampling service, topology management service) and higher-level applications.

In the present paper, we focus on the design of a large-scale churn estimator based on gossip-based aggregation [8]. Our ultimate goal is to design a component that can provide realtime estimates of churn in arbitrarily large networks. We believe this is an important first step to address the issues discussed above. We specifically concentrate our attention on churn as it is a major factor that can quickly alter the conditions of the network and have disruptive effects on the behavior of large-scale applications [18].

Our work is organized as follows. We introduce the most important contributions in the area in Section 2. Section 3 introduces our churn model, analyzes the design of the churn estimator and addresses the problem of providing accurate estimates. Section 4 presents experimental results with respect to different network settings. We conclude with final remarks and future work in Section 5.

2 Related Work

Aggregation is a key functional building block for many applications: Cortese et al. in [4] point out how DHT-based peer data management requires such functionality to provide data indexing. With respect to completely decentralized scenarios, Cai et al. [3] propose aggregation to solve the problem of global knowledge of available resources based on local information. Previous studies have defined the fundamentals to use aggregation in different contexts. Kempe et al. [12] provide the first theoretical grounds on how the information is effectively disseminated. Bawa et al. [1] employ these concepts to identify the primitives that can be straightforwardly implemented (i.e. computation of minimum, maximum, sum, count and average). Starting with Jelasity et al. [8] more importance is given to a fully decentralized implementation using gossip-based algorithms. Further studies [11, 5] concentrate on convergence properties and describe how to perform gossip-based aggregation efficiently.

The idea of using aggregation in large-scale measurement applications has been suggested in different studies. Shafaat et al. [19] describe a robust approach to estimate the network size in structured overlays using gossip-based aggregation. While network size estimates can provide insightful information on network dynamics, they are not alone sufficient to estimate churn at an appropriate level of granularity. In large dynamic networks, where joins and departures constantly compensate each other, considerable fluctuations of the network size are rare, while the impact of churn can still be significant.

Binzenhofer et al. [2] address the problem of estimating the level of churn in a network. Their approach is tailored to structured peer-to-peer networks based on a Distributed Hash Table. Finally, Gramoli et al. [7] present an algorithm more focused on detecting and assessing churn in arbitrary networks. Their analysis is most similar, in spirit, to our work. The model they use is, however, limited to overlay networks that can be represented as an undirected graph. To date, the problem of estimating churn in arbitrary networks has indeed not received much attention. Most studies focus on assessing the effects of churn and evaluate the performance of different systems under realistic or artificial churn scenarios [15, 14, 13, 20]. Other studies [18, 17, 16, 21, 6] are targeted towards designing or improving existing systems to minimize the disruptive effects of churn. We believe that estimating churn in realtime in arbitrary networks is equally—if not even more—important to foster a completely new design perspective, with applications that can handle churn reactively and in an effective way.

3 The Churn Estimator

Our analysis on the design of a churn estimator starts from establishing a formal definition of churn suitable for our study. In the literature different churn models are commonly used. In [2] the amount of time a node spends in the system (online time) and outside the system (offline time) is used to model the churn rate. This is also similar to modeling the session time and the lifetime of a node or their difference (availability time). One of the major problems with these models is that they are only applicable in particular settings. They all assume that offline nodes will rejoin the network later and that their identity can be kept track of. Another churn model is used in [10], where the churn rate is formally defined as the number of nodes that are replaced by new nodes in each cycle, assuming a cycle-driven model for gossiping. We also assume a cycle-driven model in the present paper (i.e. we model a time unit as a cycle) but, in contrast, we establish a more general definition that can be applied to any setting.

We introduce two separate definitions to model churn in a network of arbitrary size and topology: the *join rate*, defined as the ratio of nodes joining the network at a given cycle, and the *departure rate*, defined as the fraction of nodes leaving the network at a given cycle. Note that both definitions are able to capture churn in a fine-grained way and are completely independent of the network size. This is a key advantage because different join or departure rates can be compared over time and between different settings.

We now sketch the requirements for our churn esti-

mator. First off, the estimates of the join rate and the departure rate should be as accurate as possible and highly independent of the conditions of the network, i.e. the accuracy of the estimates should by no means be affected by the level of churn present in the network. Furthermore, the estimated rates should cope with every possible scenario in the lifetime of a node: entering or leaving the network, failures, crashes, restarts, etc.. Finally, estimates must be continuously updated since they are only useful if they are able to capture network dynamics with good time resolution.

To design a churn rate estimator that can be used in many different settings. we concentrate on a model that requires only very few assumptions on the nature of the underlying overlay network. More specifically, we assume that each node has knowledge of exactly C other random nodes in the network. In our prototype, we enforce this behavior by running our churn estimator on top of a reference implementation of a peer sampling service [10], whose cache size is fixed to C. Moreover, we make the assumption that each node is at any time able to determine whether each of its current C neighbors is no longer part of the network. We enforce this by monitoring the state of outgoing links in each node. Finally, we assume a system in which every node joining the network is required to contact an existing peer by means of a special *JOIN* message. This is already required in most large-scale systems during the bootstrapping phase.

To estimate the join rate and departure rate defined above, our goal is to sample their values at a given cycle and perform a run of gossip-based aggregation to compute the estimates accurately. In the following, we consider a network of initial size N, with D departures and J joins at the particular cycle considered. Our intent is to estimate the instantaneous join rate and the instantaneous departure rate as $\frac{J}{N}$, and $\frac{D}{N}$, respectively. To maintain our estimates up-to-date, we periodically restart the gossip-based aggregation algorithm using a fixed epoch length δ and lightweight synchronization mechanisms as also done in prior work [8]. To obtain fresh estimates at each cycle, we allow δ instances of the aggregation algorithm running in parallel, each one starting at a different cycle within an epoch. As a result, our churn estimator service provides continuously updated estimates of both $\frac{J}{N}$ and $\frac{D}{N}$ to the application layer.

3.1 Model

In the following, we introduce our model and prove it formally. We consider a network of size N at a generic cycle t_c, with D nodes recognized as departed at t_c. We now make the assumption that each node counts the number of neighbors departing from the network at t_c. What would happen if we used that value to initialize a distributed averaging algorithm based on aggregation starting at t_c? To address this question, let us first consider the indegree distribution of a generic node in the network at cycle t_c. From our assumptions in Section 3 we know that, for each node n_i in the network, the following equivalence holds:

$$outdegree(n_i) = C$$

In contrast, the value of $indegree(n_i)$ depends on how well the network is balanced. Nevertheless, we know that the following result holds globally:

$$\sum_{i=1}^{N} outdegree(n_i) = \sum_{i=1}^{N} indegree(n_i)$$

Therefore we can assume the following:

$$AVG_i(outdegree(n_i)) = AVG_i(indegree(n_i)) = C$$

Let us now look at the number of links pointing to a generic node d_i leaving the network at cycle t_c. We refer to this quantity as the *ghost indegree* of a departing node d_i as seen from nodes still part of the network. We indicate this quantity as $\mathbb{G}indegree_{AL}(d_i)$. We also introduce the notion of ghost indegree of a departing node d_i as seen from nodes departing from the network in the same cycle. We quantify it as the number of links pointing to d_i from any other departing node before leaving the network. We indicate this quantity as $\mathbb{G}indegree_{DEP}(d_i)$. We can finally define the ghost indegree of a departing node d_i as:

$$\mathbb{G}indegree(d_i) = \mathbb{G}indegree_{AL}(d_i) + \mathbb{G}indegree_{DEP}(d_i)$$

Under the assumption that departures do not depend on the indegree distribution and can be modeled as a random sample of the original network, we may expect at cycle t_c the following equivalence:

$$AVG_i(\mathbb{G}indegree(d_i)) = outdegree(n_i) = C \quad (1)$$

The equation above uses the idea that a random sample of a set should maintain the arithmetic mean of the original values. This is obviously more true when the number of samples is large and the distribution of values is not too skewed. We will confirm this intuition later. Let us now look at the distribution of $\mathbb{G}indegree_{AL}(d_i)$. We define $P(AL)$ as the probability of a generic node n_i still being part of the network at cycle t_c. Assuming again that departures represent a random sample of the original network, the following equation naturally holds:

$$\begin{aligned} \mathbb{G}indegree_{AL}(d_i) &= \mathbb{G}indegree(d_i) \cdot P(AL) \\ &= \mathbb{G}indegree(d_i) \cdot \frac{N-D}{N-1} \end{aligned} \quad (2)$$

In contrast, if we look at the distribution of $\mathbb{G}indegree_{AL}(d_i)$ for a given node d_i over time, we

get very different results depending on the topology management scheme used, as depicted in Figure 1. We discuss the topology management schemes examined in detail in Section 4.

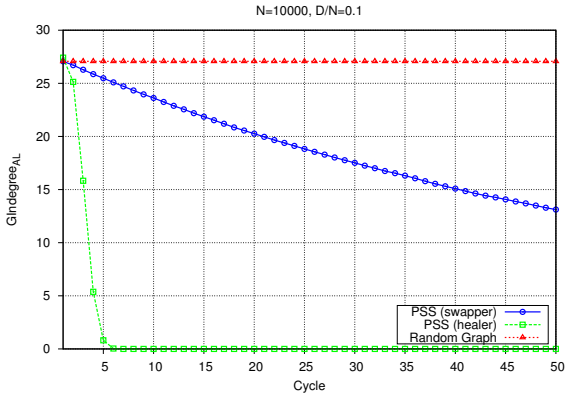


Figure 1: Comparison of the distribution of $\mathbb{G}indegree_{AL}(d_i)$ over time with respect to the adopted topology management scheme.

Figure 1 shows how fast the nodes in the network tend to forget about departed nodes. In particular we observe that it greatly varies depending on the policy adopted by the peer sampling service (PSS) (see [10] and Sec. 4 for more details). It is now clear how we can by no means make any assumption on the ghost indegree distribution of a departing node over time if we want our churn estimator to be used in different settings. Namely, our churn estimator should be possibly used in conjunction with an arbitrary topology management scheme. Since our model highly relies on the ghost indegree distribution of departing nodes, we conclude that nodes should monitor departures only on a per-cycle basis. In the most general case, we can easily achieve this effect by tagging each monitored departing node with a timestamp and checking the timestamp again in the following cycles to decide whether or not to include the node in the current set of departing nodes. Hereafter, we safely assume that only nodes departing in the current cycle are accounted by each node in its set of departing neighbors. We can now answer our first question, since we have already computed the average number of links pointing to each node departing at cycle t_c from nodes that are still part of the network. As pointed out earlier, we assume the gossip-based averaging algorithm starting at cycle t_c and being initialized with the values monitored in the same cycle. We also exclude nodes joining the network at cycle t_c from the aggregation process. Under these assumptions, the following result holds after δ cycles, namely when the aggregation process completes.

Theorem 1. *If we let each non-joining node compute* $\frac{D_i}{C}$ *(where* D_i *is the number of departing neighbors seen by a generic node node* n_i *at cycle* t_c*) and use it as the initial value of a distributed averaging algorithm, the global aggregated value* $\hat{A}_d$ *will converge to:*

$$\hat{A}_d = \frac{D}{N} \cdot \frac{N}{N-1} \tag{3}$$

Proof. If we let each non-joining node count the number of departing neighbors D_i and sum all the contributions globally, we naturally get $\sum_{i=1}^{D} \mathbb{G}indegree_{AL}(d_i)$. For linearity, if we select $\frac{D_i}{C}$ as the initial value of a distributed averaging algorithm among $N - D$ non-joining nodes, we will converge to the following global aggregated value:

$$\hat{A}_d = \frac{\sum_{i=1}^{D} \mathbb{G}indegree_{AL}(d_i)}{N-D} \cdot \frac{1}{C} \tag{4}$$

And by using the result from Equation 2:

$$\begin{aligned} \hat{A}_d &= \frac{\sum_{i=1}^{D} \mathbb{G}indegree(d_i) \cdot \frac{N-D}{N-1}}{N-D} \cdot \frac{1}{C} \\ &= \frac{\sum_{i=1}^{D} \mathbb{G}indegree(d_i)}{N-1} \cdot \frac{1}{C} \end{aligned} \tag{5}$$

As done earlier, we now assume the set of departing nodes as a good random sample of the original network. Thus, given the result from Equation 1, we get:

$$\begin{aligned} \hat{A}_d &= \frac{D \cdot AVG_i(\mathbb{G}indegree(d_i))}{N-1} \cdot \frac{1}{C} \\ &= \frac{D \cdot C}{N-1} \cdot \frac{1}{C} \\ &= \frac{D}{N-1} \end{aligned} \tag{6}$$

Therefore:

$$\hat{A}_d = \frac{D}{N} \cdot \frac{N}{N-1} \tag{7}$$

Hence, the original thesis is proved. □

The result above shows how $\hat{A}_d \approx \frac{D}{N}$ for large networks, namely when $N \gg 1$. In addition, the error of the estimate is independent of the departure rate. More specifically, we can quantify the relative error as:

$$\begin{aligned} \eta = \frac{\left|\hat{A}_d - \frac{D}{N}\right|}{\frac{D}{N}} &= \frac{\hat{A}_d}{\frac{D}{N}} - 1 \\ &= \frac{N}{N-1} - 1 \\ &= \frac{1}{N-1} \end{aligned} \tag{8}$$

As remarked earlier, we may also expect a larger error of our estimate $\hat{A}_d$ due to the approximation introduced by Equation 1. We expect our global aggregated value to match more closely the value computed

above as the fraction of departing nodes increases and the network becomes more balanced. In practice, experimental results presented in Section 4 show how we get accurate results even for unbalanced networks and low departure rates. We now turn our attention to estimating the join rate $\frac{J}{N}$. We again assume a gossip-based averaging scheme initiated at cycle t_c by every non-joining node still part of the network. In contrast, we now select a different initial value for each participant n_i. Every node n_i initializes its local estimate with J_i, i.e. the number of nodes that joined the network at cycle t_c by sending a *JOIN* message to node n_i. Under these assumptions, the following result holds once the aggregation process completes.

Theorem 2. *If we let each non-joining node compute J_i, where J_i is the number of joining neighbors at cycle t_c by node n_i, and use it as the initial value of a distributed averaging algorithm, the global aggregated value will converge to:*

$$\hat{A}_j = \frac{\frac{J}{N}}{1-\frac{D}{N}} \tag{9}$$

Proof. If we let each non-joining node count the number of joining neighbors J_i and sum all the contributions globally, we naturally obtain J. Thus, if we select J_i as the initial value of a distributed averaging algorithm among $N-D$ non-joining nodes, we will converge to the following global aggregated value:

$$\begin{aligned} \hat{A}_j &= \frac{J}{N-D} \\ &= \frac{\frac{J}{N}}{1-\frac{D}{N}} \end{aligned} \tag{10}$$

Hence, the original thesis is proved. □

The result above directly suggests a method to estimate the join rate $\frac{J}{N}$. If we assume $\hat{A}_d \approx \frac{D}{N}$ as done earlier, we get:

$$\frac{J}{N} = \hat{A}_j \cdot (1-\hat{A}_d) \tag{11}$$

We don't provide a formal expression for the relative error here, as it is dependent on the deviation of the aggregated value $\hat{A}_d$ from the actual departure rate $\frac{D}{N}$. However, we've already discussed above how we expect the error to be negligible for large networks. The model introduced in this section gives us the theoretical basis to design our churn estimator based on gossip-based aggregation. We showed how it is possible to estimate the join rate and the departure rate by monitoring local changes and performing a two-value gossip-based averaging algorithm. In the next section we then provide experimental results showing the viability of our approach.

4 Experimental Results

In this section, we present our experimental analysis carried out using the cycle-driven PeerSim simulation environment [9]. In the experiments we use the implementation of our churn estimator described in the paper[1]. Our prototype system runs on top of the reference implementation of a peer sampling service introduced in [10]. We don't explore the entire design space described in [10], but, following their findings, we constantly set the view propagation policy to *push-pull* and the peer selection policy to *tail* with the *hunting* property. We are, in contrast, interested in experimenting different view selection policies described in the paper by varying H, the self-healing parameter, and S, the swap parameter.
In the following, we show how the experimental results closely match our model and the assumptions made in the previous sections. To verify the correctness of our model independently of the topology management scheme employed, we repeat each test in three different scenarios:

PSS(healer) $H = \frac{C}{2}, S = 0$ – Peer sampling service with healer policy, i.e. keep the freshest entries.

PSS(swapper) $H = 0, S = \frac{C}{2}$ – Peer sampling service with swapper policy, i.e. minimize loss of information.

Random Graph – Static topology initialized as a random graph.

As such policies are used to produce and maintain the network topology, in our experiments we always run an initial bootstrapping phase of 100 cycles to approximate a system in a steady state (with the obvious exception of the *Random Graph* scheme).
We also test our model against several setting conditions. In the detail, Section 4.1 shows how the assumptions used in our model always hold despite variations in departure rate and network size. For lack of space, we concentrate on the departure rate, but similar results can be obtained for the join rate. Subsequently, in Section 4.2, we run experiments in similar scenarios to show how the faithfulness of our estimates is not anyhow tied to the setting. We finally conclude in Section 4.3 with an analysis on the convergence properties of our approach.

4.1 Ghost Indegree Deviation

In this section we focus on the validation of the assumptions made in Section 3.1 to develop our model. More specifically, we want to verify how departures can be effectively modeled as a random sample of the

[1] Source-code: http://www.few.vu.nl/~ortolani/files/peersim_churn.tar.bz2

network maintaining the arithmetic mean of the original indegree distribution. This assumption was used earlier to establish the following equivalences:

$$AVG_i(\mathbb{G}indegree(d_i)) = C \quad \text{and} \quad P(AL) = \frac{N-D}{N-1}$$

Validating these assumptions requires us to (i) run experiments and measure the value of the average ghost indegree and of $P(AL)$ and (ii) compare the results with the ones estimated by our model. This is equivalent to a soundness assessment of the following equation by calculating the error committed.

$$AVG_i(\mathbb{G}indegree_{AL}(d_i)) \quad = \quad C \cdot \frac{N-D}{N-1} \quad (12)$$

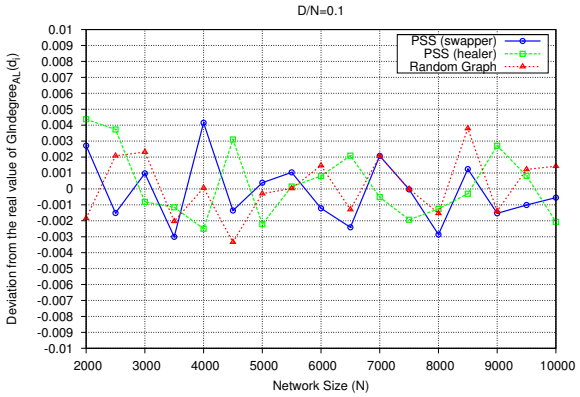


Figure 2: Deviation from the real average value of $\mathbb{G}indegree_{AL}(d_i)$ computed against different network sizes.

To this end, we report in Figure 2 the deviation of our original estimation of $AVG_i(\mathbb{G}indegree_{AL}(d_i))$ against different network sizes. In Figure 4, we finally analyze the deviation in three different departure rate scenarios:

Heavy Departure rate within the range $[0.10, 0.55]$.

Moderate Dep. rate within the range $[0.01, 0.10]$.

Light Departure rate within the range $[0.001, 0.01]$.

As documented by the figures, experimental results show how the assumptions behind our model are solid. More specifically, our original estimation of $AVG_i(\mathbb{G}indegree_{AL}(d_i))$ always shows a negligible error (in the order of magnitude of 10^{-3}) and is not anyhow dependent of the network size (Figure 2) or the real departure rate in the network (Figure 4). Although oscillatory, the error is always bounded.
No notable differences are worth reporting about the different view selection policies adopted by the peer sampling service: the only exception is *PSS(swapper)* which makes, as expected, our approach more steady with increasingly large network sizes.

Finally, we point out how our assumptions still hold even in the worst case scenario depicted in Section 3.1: unbalanced networks and low departure rates. Figure 4(c) shows how the error is still negligible for very low departure rates and using *PSS(healer)* as topology management scheme that is known to produce poorly-balanced networks [10].

4.2 Accuracy of Estimates

In the previous section we focused on validating the assumptions behind our model. The following test cases now aim to establish the quality of the estimates of the departure rate compared to the real values. This means assessing the validity of the result derived from Theorem 1:

$$\hat{A}_d \approx \frac{D}{N}$$

As done earlier, we test our model against variations of the departure rate (heavy, moderate and light), network size, and topology management scheme.
We start our validation by analyzing Figure 5, depicting the value of the estimates against different departure rates. The results confirm the validity of our model: in each scenario, our approach is able to estimate a value closely matching the real one with a very negligible error. A slightly more noticeable error (in the order of 10^{-3}) can be observed with very low departure rates. This confirms our original intuition, as discussed in the previous section.

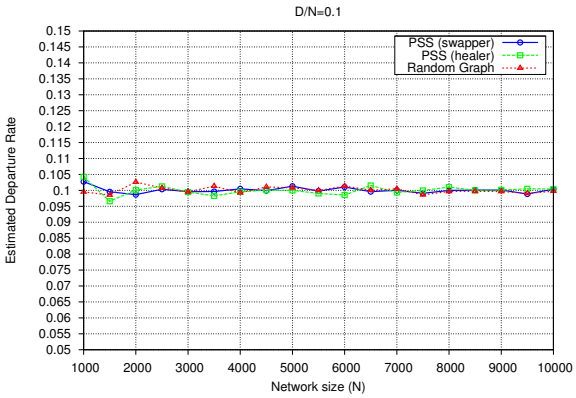


Figure 3: Estimated $\frac{D}{N}$ against different network sizes.

Figure 3 shows another important property of our approach: the estimates are highly independent of the current network size. We have experienced negligible oscillations only with very small values of N. This behavior reflects the approximation ($\frac{N}{N-1} \approx 1$) used in our model to compute the estimate of the departure rate (see Theorem 1).
To conclude, we also confirm how the topology management scheme does not affect the reliability of our estimates. Thus, our approach can be effectively used

in several different settings. This property perfectly matches our original requirements.

4.3 Convergence

We use the same approach adopted in [8] to study the convergence of our algorithm: for each cycle, we compute the normalized variance of the distribution of values possessed by all the nodes in the network. More formally, at cycle i we calculate the normalized variance y_i as:

$$y_i = \frac{\sigma_i^2}{\sigma_0^2} \tag{13}$$

We now present experimental results for a network size of 10^4 and a departure rate of 0.1. Similar trends can be observed in different settings. As also documented in other studies on gossip-based aggregation, Figure 6 shows how our gossip-based averaging algorithm converges to very accurate results in just a few cycles.

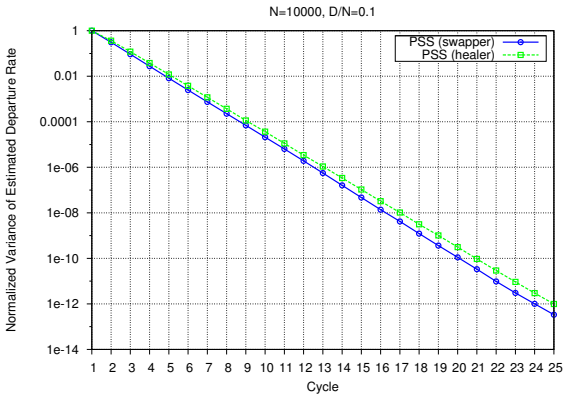


Figure 6: Normalized variance (logarithmic scale) of $\frac{D}{N}$ in a run of the gossip-based averaging algorithm.

We want to stress out how the convergence is not affected by the strategy the peer sampling service is adopting: both policies *PSS(healer)* and *PSS(swapper)* show almost identical convergence trends.

5 Conclusions and Future Work

In this paper we have presented a churn estimator based on gossip-based aggregation. The service has been designed to run on top of a generic peer sampling service implementation, while requiring very few assumptions on the network setting and being suitable for a broad range of application scenarios. The churn model adopted is very generic but fine-grained: we estimate the join rate and departure rate separately and the definitions we use are completely independent of the network size. We have introduced a model to determine the initial values of the gossip-based aggregation algorithm and formally prove the high accuracy of the resulting estimates. We have also presented a prototype based on the ideas described in the paper and showed that experimental results match closely our theoretical analysis. Experimental evidence confirmed that our approach is resilient to variations in setting conditions, such as topology management scheme, network size, and departure rate.

In future work, we are planning to evaluate our model in more realistic scenarios using a trace-driven approach, with multiple instances of the aggregation algorithm running in parallel and restarted periodically. Evaluating our approach in real-world scenarios is important to assess the accuracy of our aggregation algorithm in presence of a significant level of churn in the network. Jelasity et al. [8] present promising results in this respect, showing that the estimates provided by a gossip-based averaging algorithm are fairly accurate even in overly pessimistic churn scenarios and for extremely skewed initial distributions. In addition, we are planning to extend our analysis to scenarios where the departures cannot be assumed as a random sample of the original network and, specifically, are not completely independent of the indegree distribution. Examples include application domains where node failures can occur as a consequence of distributed denial-of-service attacks or resource-limited environments where the probability of crash for a node could possibly be related to the number of its ingoing links. We finally remark how, in our vision, the design of a churn estimator should be considered as the first ground stone to solve the problem of network awareness. We therefore plan to explore the possibility of using realtime network sampling to improve existing gossiping protocols and applications and ensure a better level of isolation between layers. Examples include: an adaptive peer sampling service, adaptive topology management, and QoS-aware protocols and applications that can better adapt to the conditions of the network and degrade service gracefully when necessary.

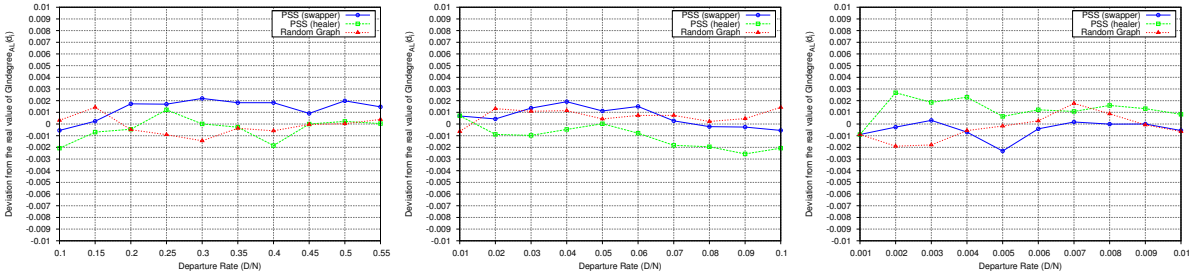
References

[1] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani. Estimating aggregates on a peer-to-peer network. Technical report, Computer Science Department, Stanford University, 2004.

[2] A. Binzenhöfer and K. Leibnitz. Estimating churn in structured P2P networks. *Lecture Notes in Computer Science*, 4516:630–641, 2007.

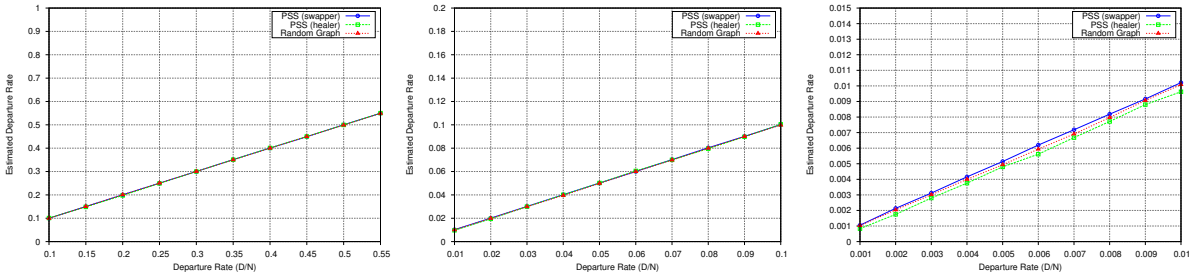
[3] M. Cai. *Distributed indexing and aggregation techniques for peer-to-peer and grid computing*. PhD thesis, University of Southern California, 2006.

[4] G. Cortese, F. Morabito, F. Davide, A. Virgillito, R. Beraldi, and V. Quema. Data aggregation in large scale distributed systems. *Emerging Communication: Studies in New Technologies and Practices in Communication*, 8, 2006.



(a) $N = 10^4$ and a heavy departure rate. (b) $N = 10^4$ and a moderate departure rate. (c) $N = 10^4$ and a light departure rate.

Figure 4: Deviation from the real average value of $\mathbb{G}indegree_{AL}(d_i)$ computed against different departure rates.



(a) $N = 10^4$ and a heavy departure rate. (b) $N = 10^4$ and a moderate departure rate. (c) $N = 10^4$ and a light departure rate.

Figure 5: Estimated $\frac{D}{N}$ against three different departure rates.

[5] L. Feng and C. Liang-Tien. Gossip-based computation of average: a closest point search approach. *Proceedings of the 2007 International Conference on Sensor Technologies and Applications (SENSORCOMM '07)*, pages 578–583, 2007.

[6] P. Godfrey, S. Shenker, and I. Stoica. Minimizing churn in distributed systems. *Proceedings of the 2006 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 147–158, Jan 2006.

[7] V. Gramoli, A.-M. Kermarrec, and E. L. Merrer. Distributed churn measurement in arbitrary networks. *Proceedings of the 27th ACM symposium on Principles of distributed computing (PODC '08)*, pages 431–431, 2008.

[8] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems (TOCS)*, 23(3):219–252, 2005.

[9] M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris. The Peersim simulator. `http://peersim.sf.net`, 2006.

[10] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *ACM Transactions on Computer Systems (TOCS)*, 25(3):8, 2007.

[11] S. Kashyap, S. Deb, K. V. M. Naidu, R. Rastogi, and A. Srinivasan. Gossip-based aggregate computation with low communication overhead. *Proceedings of the 12th International Telecommunications Network Strategy and Planning Symposium*, pages 1–6, 2006.

[12] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS '03)*, 0:482, 2003.

[13] S. Krishnamurthy, S. El-Ansary, E. Aurell, and S. Haridi. A statistical theory of chord under churn. *Lecture Notes in Computer Science*, pages 93–103, Jan 2005.

[14] J. Li, J. Stribling, T. Gil, R. Morris, and M. Kaashoek. Comparing the performance of distributed hash tables under churn. *Lecture Notes in Computer Science*, pages 87–99, Jan 2005.

[15] J. Li, J. Stribling, R. Morris, M. Kaashoek, and T. Gil. A performance vs. cost framework for evaluating dht design tradeoffs under churn. *Proceedings of the 24th IEEE International Conference on Computer Communications (INFOCOM 2005)*, 1:225–236, Jan 2005.

[16] Z. Liu, R. Yuan, Z. Li, H. Li, and G. Chen. Survive under high churn in structured p2p systems: Evaluation and strategy. *Lecture Notes in Computer Science*, pages 404–411, Jan 2006.

[17] Y. Qiao and F. Bustamante. Elders know best: Handling churn in less structured p2p systems. *Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P '05)*, pages 77–86, 2005.

[18] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a dht. *Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 10–10, Jan 2004.

[19] T. M. Shafaat, A. Ghodsi, and S. Haridi. A practical approach to network size estimation for structured overlays. *Lecture Notes in Computer Science*, 5343:71–83, 2008.

[20] D. Stutzbach and R. Rejaie. Understanding churn in peer-to-peer networks. *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 189–202, Jan 2006.

[21] Y. D. Wu and K. Ng. Analytical study on improving dht lookup performance under churn. *Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing (P2P '06)*, pages 249–258, Jan 2006.