Brief Announcement: Distributed Churn Measurement in Arbitrary Networks

Vincent Gramoli
EPFL & Univ. of Neuchâtel
vincent.gramoli@epfl.ch

Anne-Marie Kermarrec
INRIA
akermarr@inria.fr

Erwan Le Merrer
INRIA
elemerre@irisa.fr

Categories and Subject Descriptors
C.2.4 [**Computer-Communication Network**]: Distributed Systems - *distributed applications, network operating systems*;
General Terms: Algorithms
Keywords: Churn, Dynamics, Gossip, Large-Scale

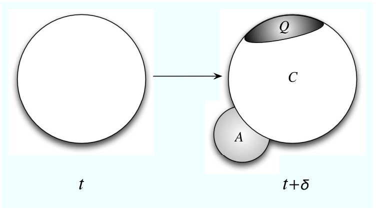
1. INTRODUCTION & MODEL

We adress the problem of estimating in a fully distributed way the dynamism over a network, called the *churn*. This BA presents, as far as we know, the first distributed method for monitoring churn in arbitrary networks, subject to arbitrary node departure and arrival patterns.

We modelize the network as follows. The system $\mathcal{G} =< V, E >$ contains N nodes, each node being uniquely identified. Each node communicates directly with its *neighbors*, V_i (degree of i is d_i). $\mathcal{G}$ is undirected, communications are synchronous and reliable, and we assume a node is able to decide whether one of its neighbor is alive (failure detection). A node knows the degree of each of its neighbors. The system is dynamic in that all nodes may fail, leave, or join the system at arbitrary time. Nodes fail on a fail stop model.When a node joins the system, it sends a joining request to one of nodes already active in the system.

To allow communication links reorganisation (i.e. not counting them as node departures or joins), a node i that drops a communication channel with a neighbor j first sends a `REORG` message to j. Thereby a link loss is considered as a failure if such message is received from the associated neighbor. Likewise, i sends a `REORG` message to its new contacts.

Objectives. Let's look at an arbitrary network at time t and $t+\delta$; let's Q be the set of nodes that were in $\mathcal{G}$ at time t but not at $t+\delta$, A the set of nodes that have joined, and C the nodes that have neither departed nor arrived. Our aim is to estimate ratios $\frac{card(A)}{N_t}$ and $\frac{card(Q)}{N_t}$ (N_t being the size of $\mathcal{G}$ at time t), that is the rate ($\in [0,1]$) of node arrivals and departures per time unit δ.



2. CHURN MONITORING ALGORITHM

The algorithm consists of two consecutive phases.

Measurement phase. All nodes monitor their neighborhood during a period δ. A node i computes its *local departure rate* as follows: if one of its neighbors, say j, quits $\mathcal{G}$, then node i increments its local counter $q_i \leftarrow q_i + \frac{1}{d_j}$. A departure is taken into account just once by addition, as d_j nodes increment their counter. In the meantime, node i counts the number of join requests it has accepted during the same period δ and increments its *local arrival rate* $a_i \leftarrow a_i + \frac{1}{d_j}$ upon connection of node j. Here d_j is j's desired degree at the end of its bootstrap process.

Averaging phase. After $t+\delta$, all nodes of C have a local records of events that happened in their neighborhood. Second phase consists in averaging all system nodes' rates, to obtain the target estimates $\frac{card(Q)}{N_t}$ and $\frac{card(A)}{N_t}$. We then use a gossip-based variance reduction algorithm [1, 2] that produces on all nodes average $\bar{a} = \frac{1}{N}\Sigma_i^N a_i$ of local counters (resp. $\bar{q}$). Due to churn, we actually compute $\frac{card(\hat{Q})}{C}$ (resp. $\hat{A}$), which is related as by definition $N_t \geq C$, and $card(Q) \geq card(\hat{Q})$.

Algorithm 1 Churn measurement executed at node i

```
1:  Measurement phase:
2:    Periodically:
3:      for all $j \in V_i$ do
4:        if downNode(j) & !receptionMsgREORG(j) then
5:          $q_i \leftarrow q_i + \frac{1}{d_j}$
6:          $V_i \leftarrow V_i \setminus \langle j, d_j \rangle$
7:          send $d_i$ to $V_i$
8:    Upon a connection of a node $j$:
9:      $V_i \leftarrow V_i \cup \langle j, d_j \rangle$
10:     send $d_i$ to $V_i$
11:     if !receptionMsgREORG(j) then
12:       $a_i \leftarrow a_i + \frac{1}{d_j}$
13: Averaging phase: Push/Pull algorithm [1]
14:   Periodically:
15:     send $\langle a_i, q_i \rangle$ to randomly selected node $j$
16:     upon receive of $\langle a_j, q_j \rangle$ from $j$
17:     $a_i \leftarrow (a_i + a_j)/2$ & $q_i \leftarrow (q_i + q_j)/2$
18:   Upon receive of $\langle a_j, q_j \rangle$ from $j$:
19:     send $\langle a_i, q_i \rangle$ to $j$ and average (as on l. 23)
```

Complexity. Measurement phase: $[cost\_failure\_detector] * 2E$, Averaging phase with *push/pull* algorithm: $2N * \#rounds$. This represents a constant number $O(1)$ of messages per node/round, leading to algorithm scaling.

[1] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3):219–252, 2005.

[2] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *In Proc. of the 44th Annual IEEE Symposium on Foundations of Computer Science*, 2003.