Self-stabilizing Power-law Networks

Thamer Alsulaiman
The University of Iowa
Iowa City, USA
thamer-alsulaiman@uiowa.edu

Andrew Berns
The University of Wisconsin – La Crosse
Wisconsin, USA
aberns@uwlax.edu

Sukumar Ghosh
The University of Iowa
Iowa City, USA
sukumar-ghosh@uiowa.edu

ABSTRACT

Power-law graphs model the interconnections in various types of large-scale networks ranging from physical and biological systems to man-made social networks and web graphs. In these graphs, the degree distribution of the nodes obeys the power-law property: the fraction of nodes $P(k)$ having a degree k closely follows the rule $P(k) \propto k^{-\gamma}$. In the domain of man-made systems, if the topology of a power-law network gets altered due to failures or adversarial attacks, then remedial actions to restore the power-law property are very important. This paper presents self-stabilizing algorithms for maintaining the power-law property in a network of processes. These algorithms allow spontaneous restoration of the power-law property from any initial connected configuration. The algorithms consist of three modular components: a detection component to detect the violation of the power-law property, an interim topology creation component, and a repair component to build the final graph. We propose two different interim topologies, a *clique* and a *linear graph*. We then present two different techniques for rebuilding the power-law topology – a probabilistic approach based on the preferential attachment model, which stabilizes in $O(\log n)$ communication rounds with a link complexity of $O(n)$ per process, and a deterministic approach that introduces the novel data structure Bridge Tree and stabilizes in $O(n)$ communication rounds with a lower link complexity.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: [Distributed networks, Network topology]

General Terms

Algorithms

Keywords

Distributed Algorithms, Self-Stabilizing, Power-Law Networks, Distributed Data Structures

ICDCN '15 Jan. 4-7, Goa, India

1. INTRODUCTION

Large-scale real-world networks have witnessed significant growth in recent years, which has prompted researchers to take a closer look at their topologies, and the mechanisms of generating these topologies. Neither regular networks like rings, grids, and hypercubes, nor the Erdős-Rényi random graphs are suitable for modeling real-world complex systems. *Small-world networks* [5] provide an acceptable model for a fraction of complex networks, but they fail to capture the topological properties of another class of graphs that are abundant in the physical and biological world, as well as in the man-made world. These include protein-protein networks, co-authorship networks, web graphs, and airline networks. One prominent characteristics of these networks is their heavy-tailed distribution of the node degrees, characterized by a power-law property: The fraction of nodes $P(k)$ having degree k closely follows the rule $P(k) \propto k^{-\gamma}$, where the value of the exponent γ typically ranges between 2 and 3. Such networks are called *power-law networks*.

An interesting problem is to explore algorithms for controlling the topology of these networks. How can an agent induce actions to alter the power-law topology of a given network over time, or restore the power-law topology from an altered configuration? In the man made world, we must be ready to cope with malicious agents that can tamper with the network or change the system behavior, thereby destroying or corrupting the topology to an extent that desirable properties like power-law are compromised. Our work raises the issue of restoration of the property instead of simply its observation.

A de-facto requirement for complex systems is the ability of self-management. The version of self-management addressed in this paper is *self-stabilization* [2], where following an adversarial action that can alter the topology of the network, the network spontaneously recovers to a desired topology (call it a *legal configuration*), and continues its legal configuration under normal actions, until the adversary hits again. We assume that adversarial actions can arbitrarily corrupt the local states of the nodes (which includes their local neighborhoods) leading the system to an illegal configuration, but such actions can never partition the system, since that makes recovery impossible to achieve. In the context of self-stabilization, the interesting feature of the current problem is that the legal configuration corresponds to a statistical distribution. In this paper, we propose the first self-stabilizing algorithm for power-law graphs.

In the man-made digital world, many *overlay networks* (networks where the links are logical links built on physical

links) are power-law networks. For instance, the Gnutella file-sharing network exhibits a degree distribution very similar to a power-law graph. Since such networks are vulnerable to adversarial attacks, research on topological self-stabilization has gained traction in recent years. However, the self-stabilization of unstructured overlay networks like Gnutella did not receive much attention since the legal configuration for such networks is not well-defined. The bulk of the effort has so far been directed to structured overlay networks that have precise structural properties. Important early work in this area addressed the LINEAR [4], which was followed up by the transitive closure-based recovery mechanism [1]. To the best of our knowledge, the only effort to stabilize unstructured (or at least not completely structured) overlay networks is due to Kniesburges et al. [3], where the authors designed a self-stabilizing small-world network that restores the small-world property from any connected configuration in linear time. The stabilization of power-law networks has remained an open problem. This paper attempts to fill this gap.

1.1 Contributions

This paper presents the first self-stabilizing algorithm for power-law network construction. The algorithm is presented as a series of modules that can be combined to reach the desired network:

1. First, we present an exploration algorithm that can detect violations in the power-law property of a given network in $O(\frac{\log n}{\log \log n})$ rounds.

2. Next, we present algorithms for building two *interim topologies* – topologies that are built solely to make it easier to rebuild the desired power-law network. The interim topologies serve as scaffolds for building the final topology. We analyze the running time and link complexity of these algorithms.

3. Finally, we present two different algorithms for constructing a power-law network: one uses a variation of the preferential attachment algorithm, and the other uses a new topology, the BRIDGE TREE. We show how the selection of the interim topology impacts the time and link complexity of these two topologies. Specifically, we show that the choice of a clique as the interim topology helps rebuild a power-law network in $O(\log n)$ rounds using $O(n)$ link complexity, whereas the choice of a linear graph as the interim topology takes $O(n)$ steps to rebuild, but with a much lower link complexity.

1.2 Organization of the paper

This paper is organized as follows. Section 2 introduces the model and the notations. Section 3 discusses the detection mechanism of our algorithm. Section 4 discusses the interim topology modules, while Section 5 discusses the repair modules. Section 6 provides a few concluding remarks.

2. BACKGROUND

Let $G = (V, E)$ denote the topology of an undirected network, and $\delta(v)$ denote the degree of the node $v \in V$. The *degree* of a node v is the number of edges incident on v. Let $P(k)$ denote the fraction of nodes in G with a given degree k. By definition, the initial configuration of G is always connected.

DEFINITION 1. *G is a* power-law graph *if and only if $P(k) \sim C.k^{-\gamma}$, where C and γ are constants.*

Note that typically $2 \leq \gamma \leq 3$. Power-law graphs have a heavy-tailed distribution – only a few nodes have very high degrees and a very large number of nodes have very small degrees. For the sake of this paper, we assume that other than satisfying the power-law property in Definition 1, no other external constraints are binding on the final topology.

Each node can execute many different kinds of actions depending on the application it supports, but our focus is only on those actions that influence the topology of G. The topology is dynamic, since new nodes join the network, or existing nodes leave the network on a continuous basis.

It has been shown that when new nodes join the network using the policy of *preferential attachment*, the resulting graph obeys the power-law property. Preferential attachment is defined as follows:

DEFINITION 2. *1. A graph with a single node is a trivial power-law graph.*

2. When a new node u joins power-law graph $G = (V, E)$ by attaching itself to m (or n if $n > m$) distinct nodes in V such that the probability of attaching to a node $v \in V$ is proportional to the degree of node v, the resulting graph satisfies the power-law property.

We use the term *node* and *process* interchangeably. Processes communicate using messages which may be of arbitrary but finite size. We assume all messages are reliably delivered, and an adversary cannot corrupt a message after it is sent. Computation progresses in rounds. In each round, a process can send a message of arbitrary size to its neighbors and receive messages of arbitrary size from its neighbors. The local state of a process consists of the values of its local variables. Depending upon the local state and the messages received in the current round, each process, in the following round, executes actions that update zero or more of its local state variables, and sends zero or more messages to its neighbors. We assume that the edges connecting pairs of processes have zero capacity, and messages reach their destination in the same round. For overlay networks, the neighborhood of a process is a component of its local state, and actions that update the neighborhood modify the network topology.

In our complexity analysis, we will fine tune the metrics of space and time complexities. In the analysis of time complexity, actions that exclude interprocess communication will be called *computation steps*. In contrast, when an action involves interprocess communication across one or more overlay links, the collection of such actions by every eligible process constitutes a *communication round*. Since communication is much slower than local computation, communication rounds are more expensive compared to computation steps, and will serve as the dominant metric of time complexity. In the domain of space complexity, *link complexity*, which counts the number of overlay links that a process needs to maintain in the worst case, will dominate the space requirements compared to the non-overlay state. This can be easily justified from the fact that there is extra work involved in maintaining each overlay link (e.g. heartbeat pinging).

3. VERIFYING THE POWER-LAW TOPOLOGY

Let G_m^n denote a random power-law graph with n nodes. The construction starts from a single node, and the n^{th} step allows a node to connect with G_m^{n-1} via m edges using the principles of preferential attachment. The class of power-law graphs generated by this approach satisfies the condition $2 \leq \gamma \leq 3$.

LEMMA 1. *Let $m \geq 2$ and ϵ be a positive real number. Then*

$$(1-\epsilon)\frac{\log n}{\log\log n} \leq diam(G_m^n) \leq (1+\epsilon)\frac{\log n}{\log\log n}$$

It follows that the diameter of a power-law graph generated via preferential attachment is $\frac{\log n}{\log\log n}$ with high probability.

We will use Lemma 1 to verify if the diameter of a given topology exceeds the bound and thus violate the power-law property. Let an initiator node v initiate a *wave algorithm* that performs a bounded traversal up to a distance $B = \frac{\log n}{\log\log n}$, collect the set S of visited nodes, and record their degrees, followed by a convergecast back to the initiator node. The collected data can now be analyzed by the initiator to determine the degree distribution. The power-law will be presumed to have fallen apart if

- $|S| < n' - c$, where the small allowance c (selected here to be $O(\log n)$) will account for new nodes which are trying to join the network but have not yet been integrated, and n' is a lower bound of the number of nodes n; or
- $|S| \simeq n'$ but the degree distribution in the visited nodes does not satisfy the power law.

LEMMA 2. *The time complexity of the Exploration Algorithm is $O\left(\frac{\log n}{\log\log n}\right)$ rounds.*

4. INTERIM TOPOLOGIES

Once the violation of the power-law property is detected, the next major task is rebuilding the power-law graph. To achieve this goal we will first build an *interim overlay network* topology as a basis and then upgrade it to a power-law graph. This modular approach simplifies analysis, and allows the rebuilding technique to be selected based upon the problem constraints.

In this section, we discuss two interim topologies: a *clique* and a *linear chain*.

4.1 A Clique

This interim topology approach uses the transitive closure framework proposed by Berns et al. [1], where the detectors take the initiative to build a completely connected topology, and then transform it into a power-law graph. The basic idea is for each detector to multicast the news about the violation, and then execute the transitive closure process. This will guarantee that every node is aware of the violation of the power-law property quickly. This is described in Algorithm 2.

LEMMA 3. *If the network is not a power-law topology, then a completely-connected network is created in $O(\frac{\log n}{\log\log n} + \log n)$ rounds.*

4.2 A Linear Chain

An alternative interim topology, the linear chain, overcomes the $O(n)$ link complexity associated with the choice of a clique as the interim topology during the rebuilding phase. However, this comes at the expense of increased communication complexity in stabilizing the power-law network. Once every node becomes a detector, the *linearization algorithm* is invoked, whose goal is to create an overlay network that is a linear chain of processes sorted according to their identifier values. The linearization algorithm is based on the idea from Onus et al.[4]. The algorithm is local, where in each round a node sorts its left neighbors, and then its right neighbors.

Once a linear chain of nodes is formed, the node with the smallest id becomes the *leader*, and it collects the ids of all the nodes in the network. This phase is built around a convergecast mechanism down the linear chain.

LEMMA 4. *If the network is not a power-law topology, then the leader collects the identifiers of all the nodes in $O(\frac{\log n}{\log\log n} + n)$ rounds.*

Once the leader collects the ids of all the nodes in the system, it can locally build the final power-law topology using either the preferential attachment method, or the BRIDGE TREE method (discussed next). There is, however, one more final step: the leader has to send to each node v its correct set of neighbors $N(v)$ up the linear chair of nodes in the intermin overlay network.

5. REPAIR MECHANISMS

In this section, we discuss two repair mechanisms that can be used provided one of the above interim topologies is selected.

5.1 Repair via preferential attachment

Once a clique is formed, each process ranks its id among the list of all ids in the system. The process with the lowest id becomes the leader, and only that process takes up the responsibility of rebuilding the power-law topology. The rebuilding will take n local computation steps: G_m^0 is a degenerate power-law graph with only the leader node, and in the ith round, the leader attaches the node with rank i to G_m^{i-1} via m edges. Once the leader locally computes the entire topology, it communicates to each node the set of its new neighbors. Each recipient will only retain these designated neighbors, and delete all other links. This completes the rebuilding phase. Conducting the repair through a leader expedites the repair by using $O(n+m)$ local steps, where m is the number of edges, and one communication round instead of $O(n)$ communication rounds.

5.2 Repair via BRIDGE TREE

We propose a new network topology called BRIDGE TREE that satisfies the power-law property, and can be generated via a deterministic algorithm (compared to the probabilistic algorithm for preferential attachment). Constructing such a topology provides an alternative mechanism for repair.

A BRIDGE TREE is a topology derived from a balanced binary search tree by adding additional bypass links (bridges) between pairs of nodes. It can be engineered to satisfy the power-law property, and the end points of the bypass links

are determined by the parameter γ of the power-law equation. After the links are added, the root becomes the node with the largest degree, and the degrees of the nodes progressively decrease as one moves from the root towards the leaves. The diameter of a BRIDGE TREE with n nodes is $O(\log \log n)$, so there exists very short paths between any pair of nodes.

To transform a balanced binary search tree into a power-law graph, the degree of each node at given level i of the tree will be augmented by adding links to sets of nodes down the tree (i.e. closer to the leaves). This step will make the degrees of all nodes at level i identical, and these nodes represent a fraction $\frac{2^i}{n}$ of all nodes in the graph. Each node v in level i finds its neighbor set $N(v)$, as follows. To satisfy power law, let k_i be the degree of a node in level i. Since $P(k_i) = \frac{2^i}{n} = C.k_i^{-\gamma}$,

$$k_i = \left(\frac{C.n}{2^i}\right)^{1/\gamma} \quad (1)$$

Although the edges of a power-law graph are undirected, for the sake of explaining this construction, assume that each node v in level i has two kinds of edges:

- The set of incoming edges from the nodes belonging to the lower levels of the tree (i.e. from nodes closer to the root); call it $v.in$. This set will consist of edges from nodes at levels up to $i - x_i^{up}$ to node v's level i.
- The set of outgoing edges from v connecting to nodes belonging to the higher levels of the tree (i.e. nodes closer to the leaves); call it $v.out$. Ideally, the choice of $v.out$ will be limited by the constraint $v.in + v.out = k_v$. However, to keep things simple, we propose that a node v at level i connect its outgoing edges to all nodes in the subtree up to level $i + x_i^{down}$ under its sibling node v' in that level (Figure 2) – this will add robustness to the topology by preventing the possibility of graph partitioning caused by the failure of the root. This will also balance the traffic load in the network. We explain the choice of x_i^{down} and x_i^{up} in the following paragraphs.

DEFINITION 3. *For a node v, an* up-neighbor *is a node $u \in N(v)$ such that $level(u) < level(v)$. Similarly, a* down-neighbor *is a node $u \in N(v)$ such that $level(u) > level(v)$. For a node v in level i, the distance to its farthest down-neighbor is x_i^{down}, and the distance to its farthest up-neighbor is x_i^{up}.*

Given $2 \leq \gamma \leq 3$, and the leader having full knowledge of nodes' ids, a BRIDGE TREE can be constructed as follows:

1. Construct a balanced binary search tree with the *root* node at level 0.
2. Draw edges from the root (level 0) to all nodes at level i, $2 \leq i \leq \left\lfloor \frac{1}{\gamma}.\log n \right\rfloor$
3. From each node at level $i = 1$, draw edges to all nodes at level j, $(i+1) \leq j \leq \left\lfloor i + \frac{1}{\gamma}.(\log n - i) \right\rfloor$ satisfying the constraint $v.in + v.out = k_i$. These nodes must belong to the subtree under the sibling of node i. Repeat this step for every node from level 2 down to the leaves.

Once the leader node locally builds the BRIDGE TREE topology, it communicates to every node $v \in V$ the list of its neighbor set $N(v)$, following which, these nodes establish the overlay links with the correct set of neighbors, and the construction is complete.

LEMMA 5. BRIDGE TREE *is a power-law graph with diameter $O(\log \log n)$, where n is the number of nodes.*

Putting all the components discussed above yields the following result:

THEOREM 6. *The composition of the Exploration Algorithm (Section 3), an interim topology (Section 4), and a repair mechanism (Section 5) is a self-stabilizing algorithm for creating a power-law graph.*

6. CONCLUSION

In this paper we presented two different self-stabilizing solutions for maintaining a power-law network. The solutions are modular, with a detection component, an interim topology component, and a repair component. Within each solution, the final repair can be performed using either the preferential attachment method or the newly introduced BRIDGE TREE method. To the best of our knowledge, the presented algorithms are the first distributed, self-stabilizing algorithms for the problem.

The stabilization algorithm has several phases, and no initialization is required in any of the phases. However, local variables can still be corrupted while the stabilization is in progress. Should that happen in one or more phases, the progress property is never compromised, although the end result can be an incorrect topology. It will require one more stabilization cycle to generate a correct power-law topology.

An interim topology serves as a scaffold and provides an important handle to the task of rebuilding the final power-law network. While the clique and the linear chain are two extreme forms of interim topologies, it may be useful to explore other forms of interim topologies (like a binary tree) that can prevent the linear growth of the communication or the link complexities.

7. REFERENCES

[1] Andrew Berns, Sukumar Ghosh, and Sriram V. Pemmaraju. Building self-stabilizing overlay networks with the transitive closure framework. *Theor. Comput. Sci.*, 512:2–14, 2013.

[2] Edsger W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Commun. ACM*, 17(11):643–644, November 1974.

[3] Sebastian Kniesburges, Andreas Koutsopoulos, and Christian Scheideler. A self-stabilization process for small-world networks. In *IPDPS*, pages 1261–1271. IEEE Computer Society, 2012.

[4] Melih Onus, Andréa W. Richa, and Christian Scheideler. Linearization: Locally self-stabilizing sorting in graphs. In *ALENEX*, 2007.

[5] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.