# Evaluating Gossip-based aggregation

An experimental study using network emulation

Niklas Semmler <semmler@kth.se>
Jiannan Guo <jiannang@kth.se>

Supervisor: Prof. Maguire <maguire@kth.se>
KTH Royal Institute of Technology

.

**Abstract**

In this study we briefly introduce the gossip algorithm and aggregation mechanism. We address the influence of network topologies on the performance of a gossip-based aggregation algorithm. Several analytical studies are discussed. We make two hypothesis regarding the relation between algorithm performance and underlying topologies. A basic gossip-based aggregation is implemented and it's performance is determined in a network emulator. Four different network topologies serve as our conditions. We collect data out of experiments and utilize statistical method to verify our hypothesis.

**Keywords**
Gossip-based aggregation, network emulation

## 1 Introduction

With the increasing interconnection of all sorts of devices their management too becomes increasingly difficult. Large scale computer networks require solutions more scalable and reliable than what conventional centralized management can provide.

Traditionally, variables of the network, such as throughput, workload and bandwidth, are gathered through the whole network by a central management system. Information is collected from all nodes of the network via periodic polling (e.g. SNMP) and stored in a database. Apart from several apparent drawbacks such as single point failure and performance bottleneck, a centralized management system cannot scale with an increasing number of nodes. In the long run this puts mission-critical business processes at risk [11].

Distributed algorithms tackle these problems by propagating messages through the whole network in a more decentralized manner. Apart from dealing with the problems mentioned above they make global information locally accessible. Other components of the network can use this information to optimize their own behavior [3].

One variant of these algorithms is the Gossip-based aggregation. Gossip-based algorithms, also known as epidemic-style techniques [2], are used to deal with drawback of deterministic algorithms in exchange for certainties. This algorithm is capable of aggregating global information in a completely decentralized manner. They differ from other distributed algorithms such as Echo algorithm proposed by Segell [9] through their lack of structure. They do not require any structure of the network, other than an awareness of immediate neighbors. That said, gossip-based algorithms could be combined with tree-based algorithms to reach an optimization for some topologies [6].

It has previously been proven that the underlying network topology has a strong impact on the performance of gossip-based algorithm [7] [3]. Yet little work exist investigating the behavior of gossip-based aggregation in real network topologies. This study focuses on the divergence of performance in different topologies of real computer networks. To this purpose a number of networks, sharing the same number of nodes, have been chosen from an online database [5].

The remainder of the article is structured as follows. Section 2 gives an introduction to the algorithm and our methodology. Next, section 3 presents the chosen implementation. Both the emulation of the networks and the implementation of the gossip algorithm are explained. Subsequently in section 4 the results of the experiment is presented. Conclusions and future work as well as related work is highlighted in section 5.

## 2 Theory

### 2.1 Related work

Motivated by peer-to-peer and ad hoc networks, a considerable number of studies have been done regarding gossip-based algorithms. Convergence and upper bound consensus time have been proven by J.Lavaei and R.Murray in [7]. Analytical methods and simulations have been utilized to discuss the relations between performance of gossip protocols and topology of network, namely randomness, connectivity etc. As a subcategory of distributed algorithm different models, such as synchronous and asynchronous models, with or without churn, are discussed in [8]. The optimization of parameters of an asynchronous randomized gossip algorithm for faster convergence is proven to be a semi-definite problem [1].

This study is mainly based on the work of M.Jelasity, A.Montresor and O.Babaoglu [3], focusing on existing static networks of different topologies [5]

### 2.2 Gossip-based aggregation

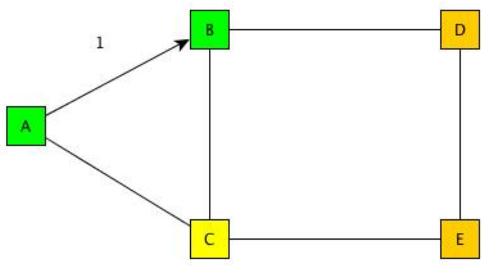
#### 2.2.1 Gossip algorithm

For better understanding of gossip algorithm, we assume a graph as in figure 1. It illustrates a typical message dissemination process applying gossip algorithm. It starts with one node containing the information to be propagated, as in subgraph (a). It randomly chooses one neighbor and send the message. In second epoch, as showed in subgraph (b), this message is passed forward to next hop in the same manner, including the root node. By recursively applying same procedures, the message will be passed through the whole network eventually. Subgraph (c) shows the occurrence of probable redundant packet from node 3 to node 2, who's already aware of the message. Because of randomness of gossip algorithm, complete dissemination can only be achieved at a certain probability within given epochs, which can be denoted as $D(x) = \int_0^x P(\xi)\mathrm{d}\xi$

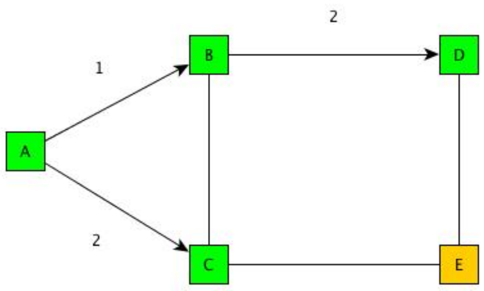
#### 2.2.2 Aggregation

A simple implementation of synchronous gossip-based aggregation algorithm, inspired by [3] can be illustrated by following pseudo code 1.

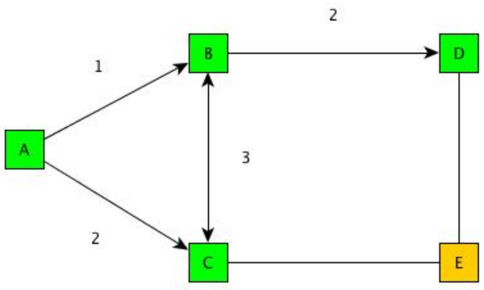
Although convergence is proven and expected convergence time can be estimated by probability density function for a certain topology [7], a drawback of probabilistic algorithm comparing to deterministic algorithm is reliability [8]. The value can only be considered



Beginning of message dissemination (a)



Passing the message (b)



Probable redundant message (c)

Figure 1: Gossip algorithm evolution epochs

```
$ActiveGossipThread$
for each consecutive $\delta$ time units at randomly picked time; do
    $q \leftarrow GET\_NEIGBOR()$
    $send\_to(state_p,\ q)$
    $state_q \leftarrow receive(q)$
    $state_p \leftarrow UPDATE(state_p,\ state_q)$

$PassiveGossipThread$
while true:
    $state_q \leftarrow receive(*)$
    $send\_to(state_p,\ q)$
    $state_p \leftarrow UPDATE(state_p,\ state_q)$
```

Listing 1: Pseudo Code for gossip-based aggregation

as true result at a certain probability. To put this protocol into practice, some extra procedures need to be added. In this study, we leave out the test of correctness inside implementation but try to obtain a empirical criteria according to the result of experiments.

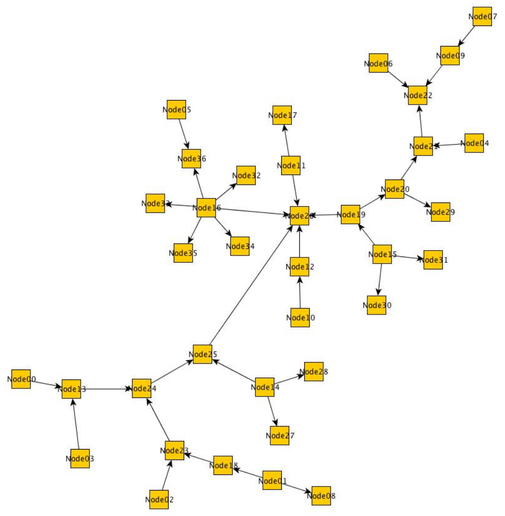
### 2.3 Impact of topology toward the performance of aggregation

A plenty of methods exist to describe overlay topologies of a network, and different representations are used to describe properties of a topology. On the other hand, the performance of gossip-based aggregation is also abstracted differently for specific purpose.

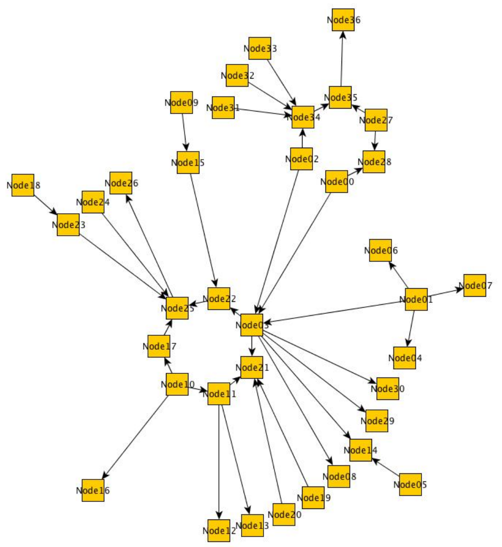
In this study, we firstly focus on investigating the impact of various number of links with the number of nodes unchanged over the performance of gossip-based aggregation algorithm.

### 2.4 Applying to real network

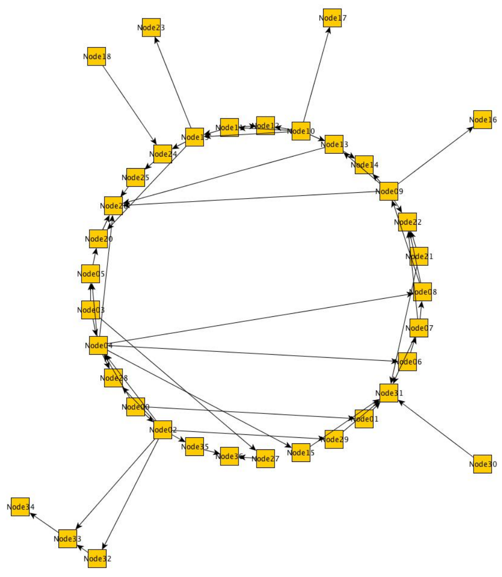
In order to apply control variable experiment method, we based our experiment on 4 real network topologies (see figure 2) with same number of nodes (37) and different number of links, as showed in Table 1.



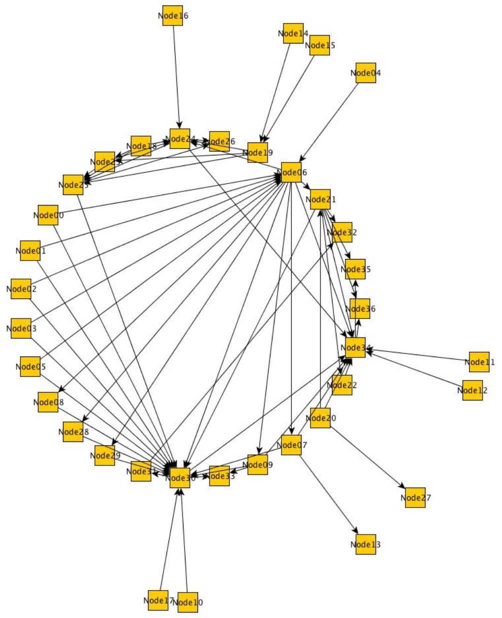
Backbone of Chile (a)



Backbone of Bulgaria (b)



European transit network (c)



Backbone network Japan, USA (d)

Figure 2: Different graph topologies (in ring layout)

### 2.5 Gossip-based aggregation

### 2.6 Hypothesis

In a given network topology $\mathcal{G} = \{\mathcal{E}, \mathcal{V}\}$, the number of links (edges) $N_E$ and convergence time $t$ are positively correlated, with number of nodes (vertices) $N_V$ unchanged.

| Name of Network | Year | Country | # of Links |
|---|---|---|---|
| Reuna | 2010 | Chile | 36 |
| BREN | 2010 | Bulgaria | 38 |
| Geant | 2010 | Europe | 58 |
| Iij | 2010 | Japen, USA | 66 |

Table 1: Topologies under investigation

### 2.7 Research methodology and methods

We are creating a positivistic investigation grounded in quantitative experiments. With induction in the form of statistical inference, we aim to predict the behavior of the system under question in the boundary of our experimental research.

We clearly distinguish ourselves from realism, as we use network graphs inspired by real networks, but leave it to other research to prove that the experiment is representative of those real networks.

Since we will investigate several existing networks with the same number of nodes (constant) but different number of links (variable), we choose experimental research as our research method.

We will apply inductive reasoning research approach to verify the hypothesis. Since our unique contribution is an investigation of real network rather than a theoretical proof, deductive reasoning research approach won't fit our goal properly.

## 3 Implementation

### 3.1 Overview

The implementation of our experiment contains three components. First, the implementation of the gossip-based aggregation is clarified. Next, the network emulation context is presented. Finally it is shown under which conditions the experiment was run.

### 3.2 Implementation of gossip-based aggregation daemon

As the gossip agent in our emulated network we developed a daemon based on the pseudo code of the gossip-based aggregation algorithm (see section 2). The programming language Python [?] was chosen for this task as it's simplicity and clarity enable a rapid development. Packets between nodes were exchanged in short lived TCP connections. The daemons on the individual nodes are started from outside the emulation system and write a log and results to the host system. After a number of epochs (successive fixed intervals) the daemons exit.

Two threads are coexisting (see figure 3). The active thread connects to randomly chosen neighbors at a epoch duration. The passive thread accepts connection all the time. Each thread update the state of the node after a successful connection. To make the state develop linearly both threads have to compete for a lock before establishing a connection. [1]

Data created this way over one graph is analyzed in R. We compare our four different topologies (mentioned above) and visualize the results as charts.

[1]Combining multithreading with networking led to some problems, especially when two nodes chose to actively create a connection with each other at the same time.

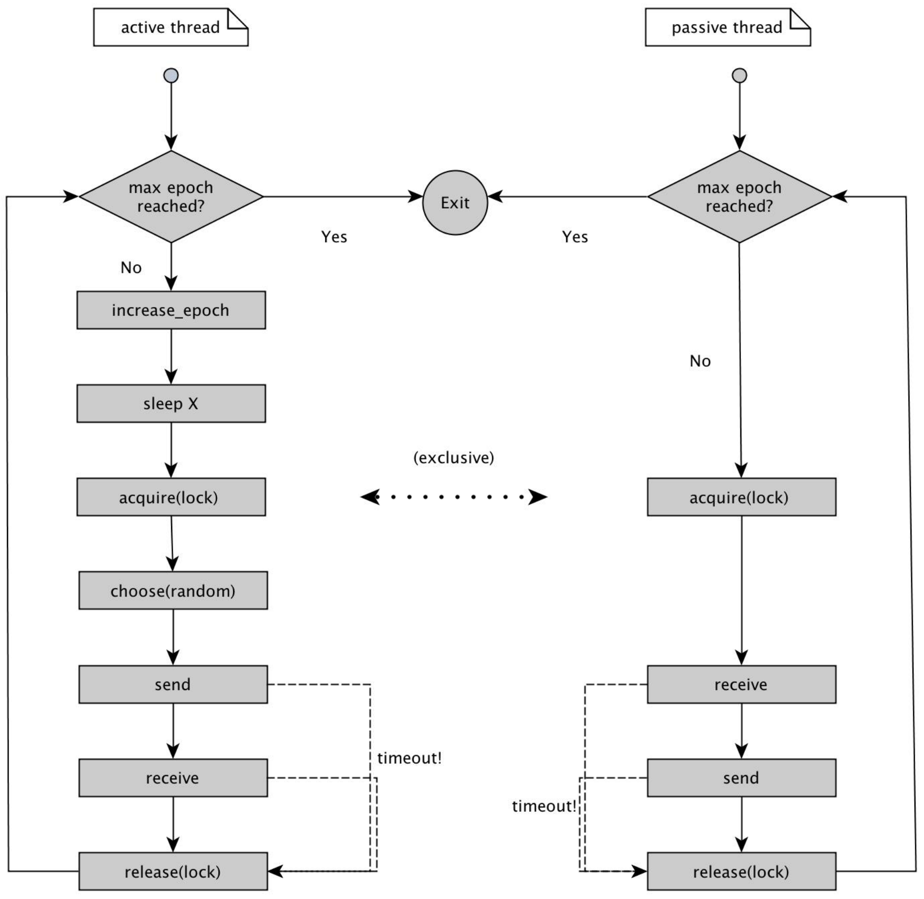


Figure 3: Flow diagram of the threads

## 3.3 Simulation of gossip-based aggregation

In a network simulation the network is simplified so as to illuminate a certain set of behaviors (e.g. Network Simulator [?]). For the purpose of simulation, gossip-based aggregation in one epoch is abstracted as a pair choosing process: in every epoch, every node is iterated exact once and one of its neighbor is randomly chosen to form a state exchange pair, and then aggregation operation, in this case add and divide by two, is applied. In this manner, convergence and different behaviors of four topologies can be predicted without running the actual emulation.

## 3.4 Emulation of gossip-based aggregation

The aim of our experiment is to evaluate gossip based aggregation for real world computer networks. To this end the gossip-based aggregation has to be implemented over a computer network. Unfortunately the setup of a real network is too time and cost intensive. Hence the next best thing was chosen: A network emulation. An emulation consists of all the elements of the real network only that the resources are not "real" but "virtual". While for example an emulated switch may process network traffic in the very same way as a switch in your office, it could be implemented as a linux process.

In this research the Netkit emulation framework (itself a wrapper to user mode linux [?]) was chosen. Netkit [?] creates virtual linux machines from a set of configuration files. These virtual machines are automatically connected through usermode processes called "uml_switch". It is possible to log on to the machines via ssh and configure them just like any other linux machines. This allows great freedom to try and test software which

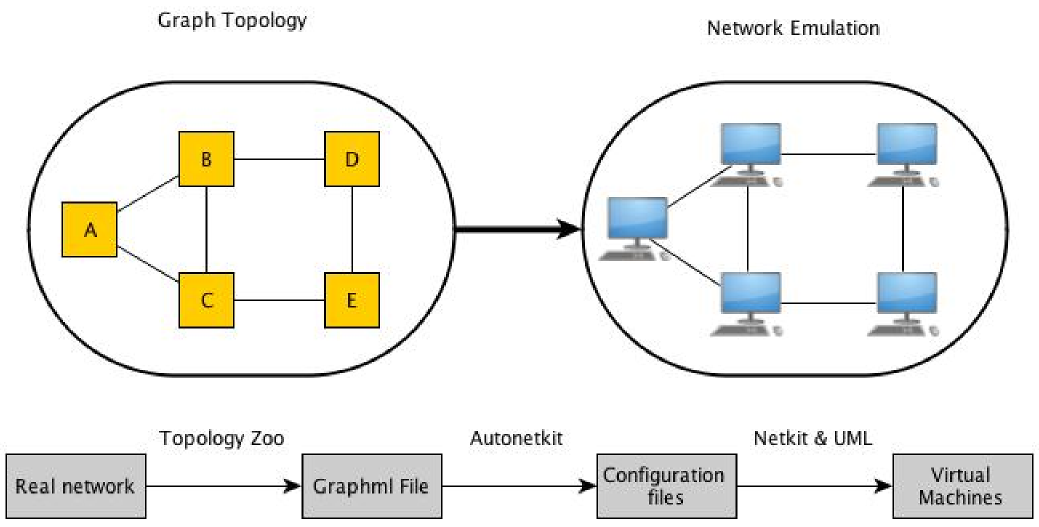


Figure 4: From a graph to an emulated network

otherwise would be hard to implement.

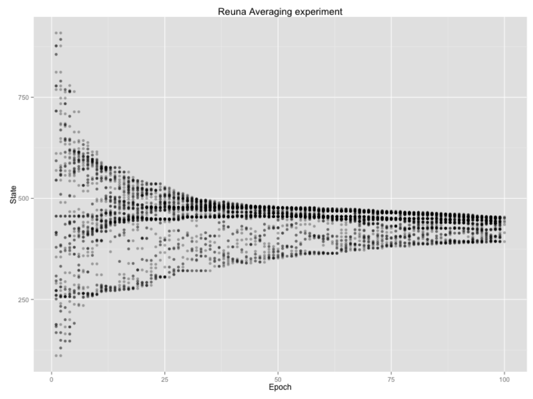
The initial set of configuration files were created with the tool Autonetkit [?] using network graphs from the "Topology Zoo" [5]. The "Topology Zoo" is a collection of around two hundred fifty graphs of telecommunication networks. The graphs are open for use and give a good idea on the reality of communication networks. Simon Knight, one of the founders of the topology zoo has developed the tool Autonetkit. It takes a graph xml file (graphml) and constructs over a series of abstractions configuration files (from OSPF to DNS). For the purpose of this experiment Autonetkit was augmented so as to create for our gossip agent a file containing all neighbors of the individual nodes.

### 3.5 Running the experiment

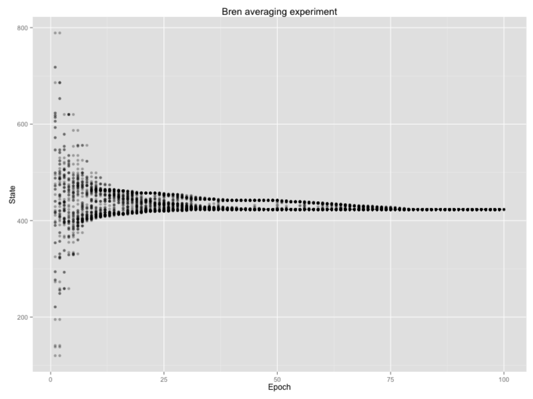
The experiment was run on a host with 64 cores, each with 2299 MHz and 126GB of memory. Each virtual machine required 32Mbs of memory. The operating system was Ubuntu 11.10.

## 4 Results & Discussion

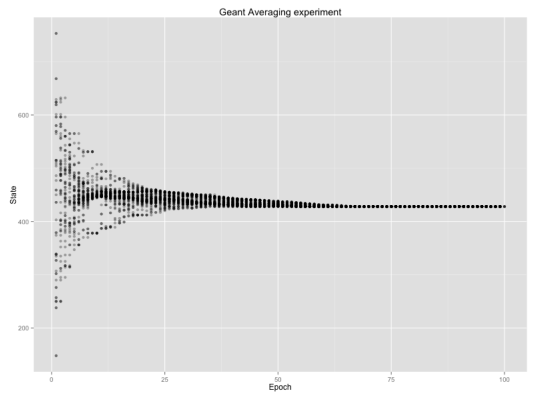
As showed in Figure 5, for each topology, state of every node in every epoch is plotted. Inside 100 epochs all four experiments are in the trend of converging and if we set convergence criteria as 5% of mean value, Bren, Geant and Iij can be considered as converged inside their run of experiments, accordingly at 75th, 60th and 40th epoch.
Thus, the order of convergence speed can be derived as Iij, Geant, Bren, Reuna, from the fastest to slowest, which verifies our hypothesis.



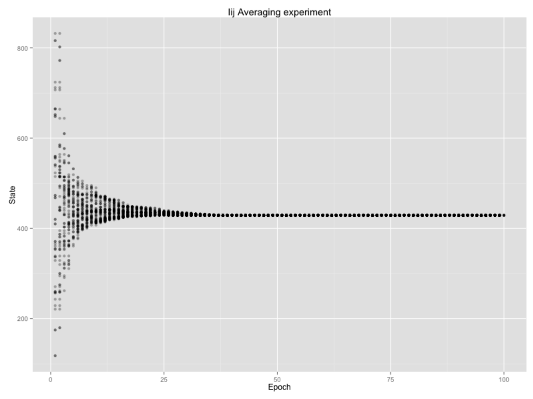
Reuna Averaging experiment



Bren Averaging experiment



Geant Averaging experiment



Iij averaging experiment

Figure 5:

## 5 Conclusion & Future work

### 5.1 Conclusion

### 5.2 Future work

#### 5.2.1 More comprehensive emulation

Since we assume an undirected unweighted static topology, some other crucial properties of a real network are not reflected in our study, such as link speed and congestion. Although, in real backbone networks, node leaving and joining are not likely to happen, but packet loss and delay could be modeled into churn mechanism and should be considered in future work.

#### 5.2.2 Relation between the performance and other properties of graph

In [3], convergence factor $E(2^{-\phi})$ is used to determine convergence time, smaller convergence factor results in faster convergence. The Watts-Strogatz is used to model the topology of overlay network, indicating randomness as an independent variable [12]. Thus, a function is derived, with randomness parameter $\beta$ as input and convergence factor $E(2^{-\phi})$ as output.
Although, to apply this model in a real network, reverse procedures need to be done to determine randomness parameter $\beta$ of a given graph $\mathcal{G} = \{\mathcal{E}, \mathcal{V}\}$. Mapping this back to the study of [3] can give a better understanding how topology impact convergence time.

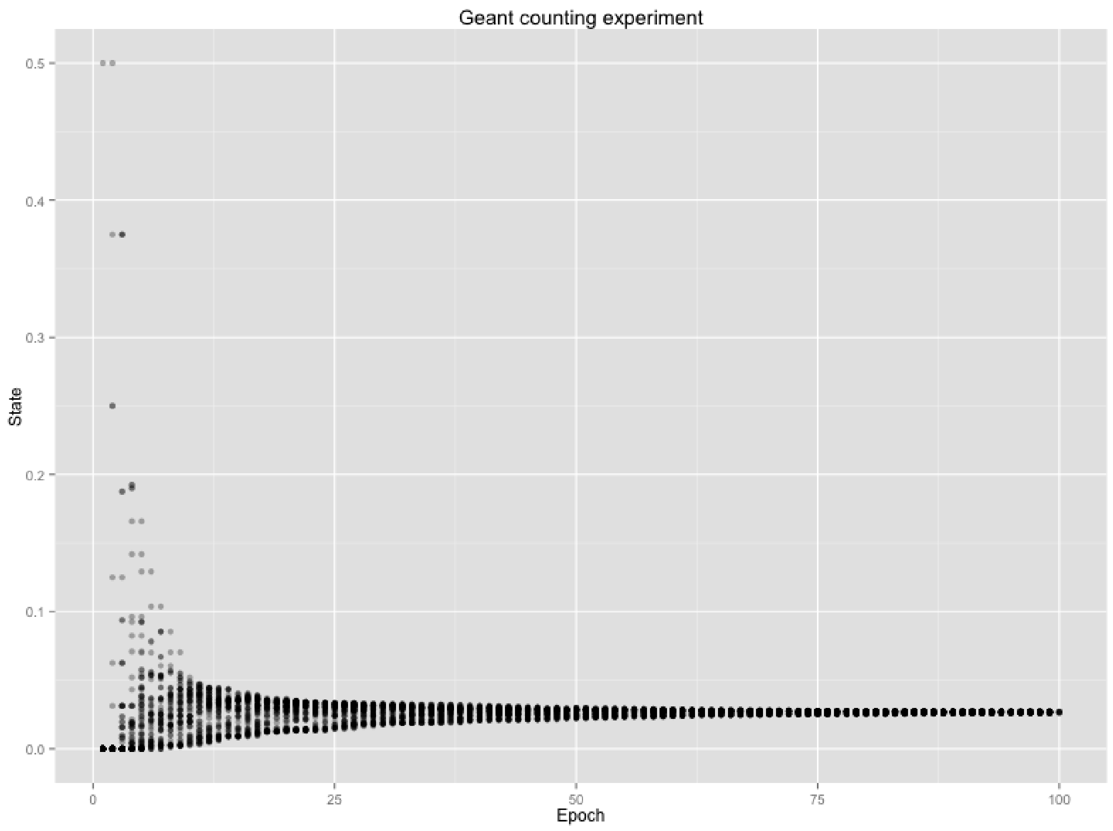


Figure 6: Geant counting

Other than this factor, graph clustering [?], max-flow and min-cut [?] could also be derived from these graphs and relationship between them and gossip algorithm performance can be further discussed.

## References

[1] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Analysis and optimization of randomized gossip algorithms. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 5, pages 5310–5315 Vol.5, 2004.

[2] Indranil Gupta, A-M Kermarrec, and Ayalvadi J Ganesh. Efficient and adaptive epidemic-style protocols for reliable and scalable multicast. *Parallel and Distributed Systems, IEEE Transactions on*, 17(7):593–605, 2006.

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

[4] Li Ji, Wang Bing-Hong, Wang Wen-Xu, and Zhou Tao. Network entropy based on topology configuration and its computation to random networks. *Chinese Physics Letters*, 25(11):4177, 2008.

[5] Simon Knight, Hung X. Nguyen, Nick Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *Selected Areas in Communications, IEEE Journal on*, 29(9):1765–1775, 2011.

[6] Pradeep Kyasanur, Romit Roy Choudhury, and Indranil Gupta. Smart gossip: An adaptive gossip-based broadcasting service for sensor networks. In *MASS*, pages 91–100. IEEE, 2006.

[7] J. Lavaei and R.M. Murray. Quantized consensus by means of gossip algorithm. *Automatic Control, IEEE Transactions on*, 57(1):19–32, 2012.

[8] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[9] Adrian Segall and Inder S. Gopal. Distributed name assignment in computer networks. *Computer Networks*, 17:127–139, 1989.

[10] Ricard V. Sole and Sergi Valverde. Information theory of complex networks: On evolution and architectural constraints. In *In*, pages 189–210. Springer-Verlag, 2004.

[11] Rolf Stadler. Protocols for distributed management. Technical Report 2012:028, KTH, Communication Networks, 2012. QC 20120604.

[12] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.