Push-gossip protocol efficiency with network topology propagation

Alexey Vanin[1,2] and Vladimir Bogatyrev[2]

[1] NEO Saint Petersburg Competence Center, Saint Petersburg, Russia
alexey@nspcc.ru
[2] ITMO University, Saint Petersburg, Russia
vabogatyrev@corp.ifmo.ru

Abstract. Effective data propagation between nodes is crucial factor in distributed systems. Reactive notifications allow responding quickly to various system events: failures, topology changes, etc. The general approach is to use the gossip protocol. Data is distributed as a rumor; each node retransmits input messages to another nodes. In this paper, we consider a particular case of the gossip protocol usage. Distributed system informs included nodes about the participation in collective challenge, thereby forming a sub-network topology. We evaluate efficiency with the simulation model of topology propagation process.

Keywords: gossip protocol, distributed system, computer network

1 Introduction

The distributed system efficiency can be evaluated by the speed of reaction to internal or external events [3,4,5]. The system takes time to make a decision and propagate instructions among nodes. Rapid data distribution preserves integrity, which is one of the systems properties [1].

Gossip protocol is used to distribute alerts over the network in distributed systems [6]. There are different approaches for gossip protocol implementation. First approach is the push-gossip protocol [6,7]. Nodes transfer data for some amount of time. ime can be chosen sufficiently high, so all participants with high likelihood will receive the data. This is called rumor-mongering protocol. Also nodes can send data on demand until it is made obsolete by newer information. This is called anti-entropy protocol. It is useful for sharing information reliably among a group of participants [2].

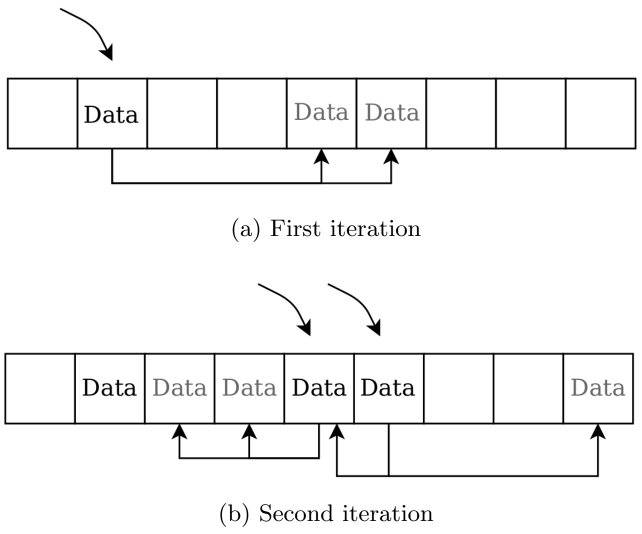
Second approach is the pull-gossip protocol [6,7]. Data is periodically requested on demand by nodes. This approach allows to decrease number of transmitted messages in the network. Pull-gossip protocol is synchronous, while push-gossip protocol can work asynchronously. Also, there is combined approach, where the system starts propagation as push-gossip and then uses pull-gossip for decreasing the load on network [6,9]. Average consensus [10] and averaging gossip algorithms [11] also form an special case of usage.

All this implementations transmit notifications and the data within the frame of established network topology [8]. Consider distributed system within transient state. Nodes often connect and disconnect from the system. System chooses set of nodes and propagate instructions among them, so they can know each other to do the set of system tasks together. There is a connection on physical layer between nodes, but gossip protocol defines logic topology of the system. Such case implies the use of push protocols. Nodes are waiting for input gossip messages with instructions to execute. However, due to the decentralized nature of the system and the lack of prior synchronization, there is an issue of determining gossip transmission parameters. In this paper we will examine the simulation model of push-gossip propagation process as a method to determine optimal transmission parameters.

2 Simulation model

Gossip propagation process can be defined by several parameters: number of participants in the process, fan-out (number of random data recipients), recipients choosing algorithm, etc. Propagation time and nodes notification rate are characteristics of the process.

The simulation model is built with the usage of general-purpose programming language (source code available at github.com/nspcc-dev/gossip-model). As an input model takes number of nodes S, fan-out size f and number of experiments. The simulation stages are schematically presented in Figure 1.



(a) First iteration

(b) Second iteration

Fig. 1. Model process representation

For every node the model generates structures and sets data flag in the first one. Simulation process consistently repeats three stages:

– planning,
– propagation,
– evaluation.

At the planning stage the model chooses data sender nodes. In this paper we examine model, which chooses nodes with data flag that has not sent any data yet. In this case, every node in the system will retransmit data only once. This simplifies real implementation of algorithm and reduces number of transmitted messages.

At the propagation stage, data sender nodes independently choose recipients. The number of recipients is defined by fan-out parameter f. This operation is based on a pseudo-random number generator with a uniform distribution. Model sets data flag on every recipient.

At the evaluation stage, the model collects statistics and checks stop condition. Stop condition is met if all nodes have data flag. In this paper it is called *saturation*. If stop condition is not met, stages are repeated again. In this paper it is called *iteration*. Figure 1a represents first iteration and Figure 1b represents second iteration. Number of iterations considered as a data propagation time.

The model outputs table with numbers of iterations and number of experiments that has been finished with exact propagation time. With precision based on number of experiments, we can determine saturation probability for constant S, f and propagation time limit as a number of iterations.

The model has a number of simplifications. Iteration is a synchronous process, but in the real system data propagates asynchronously. However, it affects the speed of propagation and does not affect the order. There is no parameter for data loss probability in data channels and the model considers only the mesh topology in the system.

3 Simulation results

3.1 General interpretation

As described in section 2, the model allows to determine saturation probability for constant S, f and the number of iteration limit. Consider iteration limit as 3 iterations and vary fan-out parameter f for different network sizes. Results are presented in Figure 2.

Saturation probability has the same curve for different S. With a low fan-out parameter f, the model has never met the iteration limit, thus the possibility of saturation has the value of 0. Then there is the area, where saturation probability grows up to the value of 1. After that, there is no point in increasing the fan-out parameter.

Data propagates faster if there is no collisions between random sets of recipients. Consider fastest propagation time for fixed fan-out size f and iteration limit

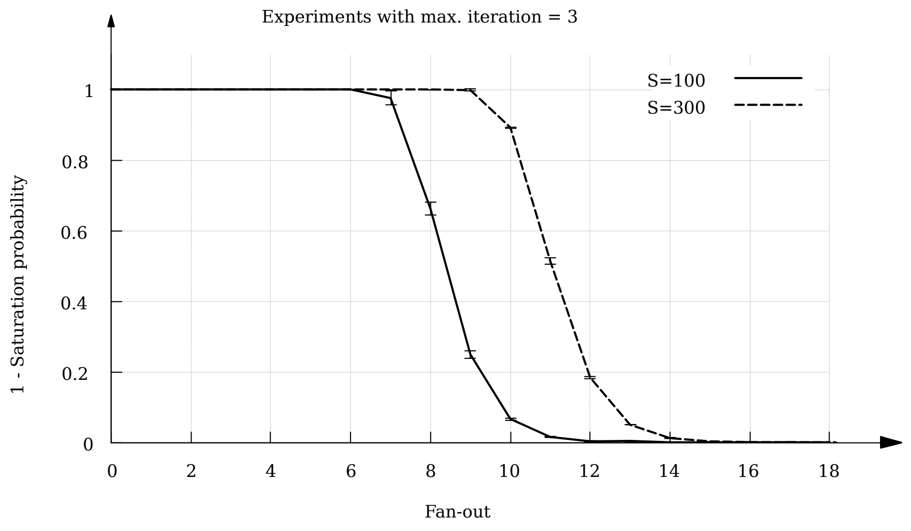


Fig. 2. Estimation of saturation probability with different fan-out

h. First node sends f unique messages, then f nodes sends f^2 unique messages. After h iteration, the number of all sent messages must be equal S:

$$S = \sum_{i=0}^{h} f^i \tag{1}$$

Parameter f defined in (1) actually is the minimal fan-out size, where saturation is still possible, even with low likelihood.

Consider the worst scenario when all data of the sender nodes choose the same recipients. In this case, saturation is possible if fan-out has a size of:

$$f = S - 1 \tag{2}$$

Parameter f defined in (2) actually is the maximum fan-out size, where saturation is possible. But probability for all pseudo-random number generators to be synchronized is extremely low.

3.2 Optimal fan-out calculation

Saturation probability function with any parameters can be represented as in Figure 3. Set of fan-out sizes lay between f_{min} and f_{max}. In real distributed system, it is reasonable to define saturation probability with expected reliability parameters. In this case, we can determine optimal fan-out for that saturation probability bound. In Figure 3 it is the intersection between probability saturation function and boundary v.

The model allows to find relation between number of nodes S and optimal fan-out f_{opt}, so distributed system may increase the scale effectiveness. We have

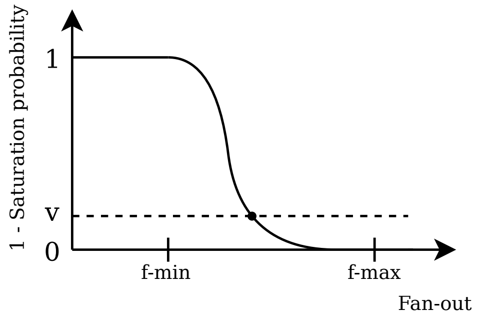


Fig. 3. Saturation probability function and optimal fan-out

modeled distributed system with iteration limit $h = 3$, saturation probability at least 0.99999 and found out optimal fan-out for different S. The obtained set of values can be expressed as a function with the logarithmic regression:

$$f_{opt}(S) = \begin{cases} 5.5 \ln S - 4.8, & \text{if } S < 30, \\ 1.4 \ln S + 9, & \text{if } S \geq 30 \end{cases} \tag{3}$$

As soon as saturation probability function consists of two convex curves, f_{opt} is also set as a pair of logarithmic functions. Overall results are presented in Figure 4.

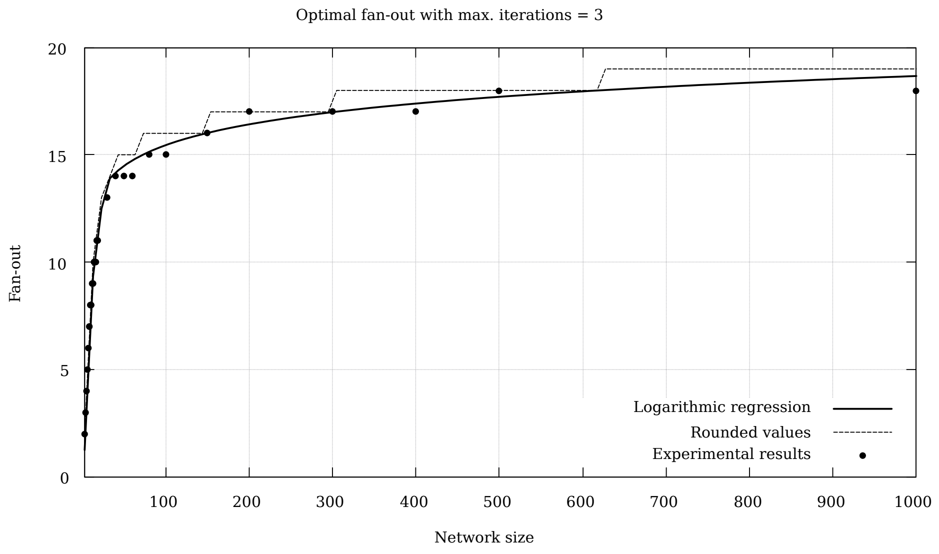


Fig. 4. Optimization task for finding optimal fan-out

4 Conclusion

The simulation model allows to evaluate the effectiveness of the push-gossip protocol for network topology propagation with different propagation parameters. With the experiments, the saturation function was determined with f_{min} and f_{max} values. In addition, the model allows to determine the optimal fan-out value f_{opt} with fixed network size S, probability saturation boundary and propagation time h. It allows to define optimal fan-out function for scalable distributed systems. In this paper we defined optimal fan-out function (3) for distributed system with propagation time $h = 3$ iterations and saturation probability at least 0.99999.

Further research includes adding other physical topologies to the model. Also adding data loss probability and examining saturation probability function with new parameters.

References

1. Aliev T.I. The basics of discrete systems modeling. St. Petersburg: ITMO, 2009. (in Russian)
2. Renesse R., Dumitriu D., Gough V., Thomas C. Efficient Reconciliation and Flow Control for Anti-Entropy Protocols. *LADIS '08 Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware*, ACM, 2008, No. 6.
3. Bogatyrev V.A. Exchange of Duplicated Computing Complexes in Fault tolerant Systems. *Automatic Control and Computer Sciences*, 2011, Vol. 45, No. 5, 268276.
4. Bogatyrev V.A., Parshutina S.A. Redundant Distribution of Requests Through the etwork by Transferring Them Over Multiple Paths. *Distributed Computer and Communication Networks*, 2015, CCIS, vol. 601, 199207.
5. Saidi A., Mohtashemi M. Minimum-cost first-push-then-pull gossip algorithm. *IEEE Wireless Communications and Networking Conference*, WCNC, 2012, 25542559.
6. Demers A., Greene D., Hauser C., Irish W., Larson J., Shenker S., Sturgis H., Swinehart D., Terry D. Epidemic Algorithms for Replicated Database Maintenance. *Proceedings of Sixth Symp. on Principles of Distributed Computing*, ACM, 1987, 112.
7. Apt K., Grossi D., Hoek W. When Are Two Gossips the Same? Types of Communication in Epistemic Gossip Protocols. 2018, URL: https://arxiv.org/pdf/1807.05283.pdf
8. Gupta, Indranil, Chandra, Tushar D., Goldszmidt, German S. On scalable and efficient distributed failure detectors. *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing*, 2001, 170179.
9. Birman K. The Promise, and Limitations, of Gossip Protocols. *SIGOPS Oper. Syst.Rev.*, 2007, 41(5), 813.
10. Fagnani F., Zampieri S. Randomized consensus algorithms over large scale networks *IEEE JSAC*, 2008, vol. 26, no. 4, 634649.
11. Boyd S., Ghosh A., Prabhakar B.,Shah D. Randomized gossip algorithms. *IEEE Trans. Info. Theory*, 2006 vol. 52, no. 6.