A Gossip Protocol for Subgroup Multicast

Kate Jenkins, Ken Hopkinson, Ken Birman

Department of Computer Science, Cornell University

{katej, hopkik, ken}@cs.cornell.edu

Contact Info:

Phone: (607) 255-9199

Fax: (607) 255-4428

Address: Dept of CS, Cornell University (Upson Hall), Ithaca 14853

Abstract:

Gossip-based multicast can be an effective tool for providing highly reliable and scalable message dissemination. Previous work has shown it to be useful in a variety of group communication settings when processes all belong to a single process group. In this paper, we consider the problem of gossiping within overlapping process subgroups. If each subgroup independently runs the standard gossip protocol, then the total gossip overhead could be high for a process that is a member of many subgroups. We present a novel gossip protocol that allows individual subgroup members to trade-off update quality for gossip overhead, enabling processes to belong to several subgroups while maintaining a low total gossip overhead. Our results include a mathematical model for message dissemination under this modified gossip protocol, and an algorithm that computes gossip parameters such that all processes within a subgroup achieve their desired update quality. Preliminary results are promising.

1 Introduction

Process groups are useful in many reliable group communication settings. A process group is a set of processes that share state, updated using multicast messages to achieve a common goal. Process groups are often found as part of a larger virtual synchrony group communication framework, where there is a view manager that manages group membership, with reliability guarantees for message delivery.

Some applications use large numbers of process groups, often with heavily overlapping membership. For example, process groups have been used for a reliable network file system and a fault-tolerant distributed object system (see [7]). People have developed light-weight group services to amortize the virtual synchrony overhead when there are large numbers of overlapping process groups [7, 5].

Another recent development in reliable group communication has been the application of epidemic rumor-mongering techniques to reliable multicast [4]. Epidemic "gossip" protocols provide strong probabilistic reliability guarantees, and are extremely scalable. Gossip protocols were first developed to improve consistency management of replicated databases [3], and have also successfully been used for failure detection [9] and membership detection [6].

A pure gossip protocol takes place in rounds, where in each round a participating process selects another uniformly at random they share information. There are a number variations on how this is done. In "push" gossip, the process that initiates the gossip "infects" the selected target with information. It has been shown that in a group of N machines, if one machine starts out with a novel piece of information, it takes $O(\log N)$ rounds for every machine to become infected with that information (with high probability) [1, 8]. These are called epidemic algorithms because the analysis of their behavior stems from work in epidemiology, studying the spead of infection within a population. In scenarios where participants are constantly generating new information, one could have gossip running continuously, with the same overhead for each participant.

We consider a modified gossip protocol that can be useful when some processes want to be members of a gossip group, but are willing to accept a tradeoff between the amount of information they receive in order to incur a smaller gossip overhead. In particular, we consider the situation where group members are interested in receiving updates generated by other members, as long as they arrive within some fixed time after being generated. We call this the timeout. Each group member can specify a "quality rating" between 0 and 1. A member with rating r wants to receive wants to incur fraction r of the gossip overhead of the other group members, and is willing to receive only fraction r of the updates. Such a protocol could be useful in a group communication setting where there is a large number of overlapping process groups, and some processes are interested in the information generated by many groups yet could not handle the gossip overhead of being full members all of the groups. This problem is motivated by an application to power networks, where processes participating in load-following contracts require very detailed data and are willing to pay for it, while other processes merely monitor the contract and require much less detail. The latter processes would prefer to pay less for lower-quality data.

We call our protocol "gravitational gossip", because it evokes the fact that messages are more likely to be both sent and received by the "heavier" members with high quality ratings within the group.

In this paper, we restrict our attention to gossiping within a single LAN. We assume that the communication graph is fully connected and that the distributions of latencies are similar on all links, and do not worry about router congestion or other issues that would need to be addressed in a real implementation.

We present a mathematical model for gravitational gossip and an algorithm that uses this model to compute the gossip parameters necessary to achieve the desired performance. We then compute the gossip overhead incurred by machines belonging to multiple subgroups at different ratings.

2 Gravitational Gossip

Gravitational gossip is similar to push gossip, except that instead of selecting one target uniformly at random each round, the frequency of target selection and the choice of target is weighted by some extra parameters. Throughout this exposition, we will use the terms "member", "process" and "machine" interchangeably.

Each machine has two parameters, its *infectivity* and its *susceptibility*. We denote these by I_m and S_m respectively for machine m. During a round of gravitational gossip, machine i gossips to machine j with probability $I_i * S_j$ (as compared to 1/N for standard gossip). All machines at the same rating will be assigned the same infectivities and susceptibilities. We say these machines belong to the same *strata*. In an abuse of notation, we will let I_r and S_r denote these values for rating r.

This model of infection spreading is referred to as *proportionate mixing* in the epidemiological literature [2]. We refer to a machine that has received the update as being *infected*, and one that has not received it yet as being *suspectible*.

Our problem reduces to finding values for I and S that achieve the desired level of infection for machines at each quality rating.

3 A Mathematical Model

We make some assumptions when developing a model for the protocol's behavior. One assumption is that there are large numbers of machines at each rating. A second is that the fraction of machines infected by each round is very close to its expected value (i.e. there is very little variance). Boris Pittel proved this to be the case for the standard push protocol when each message is gossiped independently [8], and we anticipate that this is still the case for gravitational gossip.

Given these assumptions, we can create deterministic recurrence relations for the fraction of susceptible machines at each rating as a function of time (measured in rounds).

let,

$$x_r(t) = \text{fraction of susceptible machines at rating } r \text{ at time } t$$

Note that we can also interpret $x_r(t)$ as the probability that any given machine at rating r is still susceptible by time t.

Then,

$$\begin{aligned}
x_r(t+1) &= P(\text{machine } m \text{ at rating } r \text{ is susceptible at time } t+1) \\
&= P(m \text{ was susceptible at time } t) * P(\text{no infected machines gossiped to } m \text{ at time } t+1) \\
&= x_r(t) * P(\text{each machine that gossiped to rating } r \text{ at time } t+1 \text{ missed } m) \\
&= x_r(t) * (1 - 1/N_r)^{\text{num gossips to rating } r \text{ at time } t+1} \\
&\approx x_r(t) * (1 - 1/N_r)^{N_r * S_r * \sum_j I_j * N_j * (1 - x_j(t))} \\
&\approx x_r(t) * e^{-S_r \sum_j I_j * N_j * (1 - x_j(t))}
\end{aligned}$$

Where the sum in the exponent sums over all of the different ratings, and N_r is the number of machines with rating r.

Now, let $\alpha(t) = \sum_j I_j * N_j * (1 - x_j(t))$.

Putting this all together, we get a system of recurrence relations:

$$\begin{aligned}
x_1(t+1) &= x_1(t) * e^{-S_1 * \alpha(t)} \\
x_{r_1}(t+1) &= x_{r_1}(t) * e^{-S_{r_1} * \alpha(t)} \\
x_{r_2}(t+1) &= x_{r_2}(t) * e^{-S_{r_2} * \alpha(t)} \\
&\cdots \\
x_{r_k}(t+1) &= x_{r_k}(t) * e^{-S_{r_k} * \alpha(t)}
\end{aligned}$$

This gives us our prediction for the fraction of susceptible machines at each rating at each round. The fraction of infected machines at round t is just $1 - x_r(t)$ for each rating r. Note that it depends on our choices of S_r's, I_r's, N_r's and $x_r(0)$'s.

There is one very nice property which follows directly from these recurrence relations. Namely,

$$\forall t, x_r(t) \approx (x_1(t))^{S_r/S_1}$$

This can be shown true by induction, since it gives:

$$x_r(t+1) = x_r(t) * e^{-S_r * \alpha(t)} = (x_1(t))^{S_r/S_1} * (e^{-S_1 * \alpha(t)})^{S_r/S_1} = (x_1(t) * e^{-S_1 * \alpha(t)})^{S_r/S_1} = (x_1(t+1))^{S_r/S_1}$$

and $x_r(0) \approx x_1(0)^{S_r/S_1}$ for N_1 large.

4 Model Parameter Prediction

Recall that the problem we set out solve is to have machines at rating r have probability r of receiving any given time-series update before it times out, and to have them initiate r times as many gossips as a machine at rating 1. Translated into our gravitational gossip framework, that means we want to find values $I_1, I_{r_1}, \ldots, I_{r_k}$, and $S_1, S_{r_1}, \ldots, S_{r_k}$ such that:

$$x_1(timeout) \approx 0$$
$$x_{r_1}(timeout) \approx 1 - r_1$$
$$\ldots$$
$$x_{r_k}(timeout) \approx 1 - r_k$$

and such that $I_{r_1}/I_1 = r_1, \ldots, I_{r_k}/I_1 = r_k$.

Note that from the recurrence relations, $x_1(t)$ will never equal 0 (although it will get arbitrarily close). This follows from the fact that no matter now long we wait, the probability that a given machine will become infected will never equal 1 since there is always some chance it will be missed. Instead, we let the user specify an additional parameter, δ, and find parameters such that $x_1(timeout) < \delta$.

The parameter prediction algorithm works as follows:

We know that for any fixed parameter settings, there is some smallest time T such that $x_1(T) < \delta$.

By the result of the previous section, at that time, $x_r(T) = (x_1(T))^{S_r/S_1} < \delta^{S_r/S_1}$, for each rating r. So we let $S_1 = 1$, and pick S_r such that $x_r(T) < 1 - r$. Namely, we want $\delta^{S_r} = 1 - r$, so we choose:

$$S_r = \log_\delta(1 - r) = \log_2(1 - r)/\log_2(\delta), \forall r$$

This gives us the desired fractions of infected machines at each rating at time T. However, we want this to be achieved by time $timeout$. To achieve this, we can globally scale all of the susceptibilities by some value γ. No matter what the value of γ is, whenever $x_1(t) < \delta$ the fraction of susceptibles at the other ratings will satisfy the desired proportions. Varying γ just affects the time at which this occurs. Hence, we binary search on the value of γ to find one for which $x_1(timeout) < \delta$, but $x_1(timeout - 1) \geq \delta$. We test a given value of γ by running the recurrence relations with the corresponding parameter values.

This algorithm runs quickly, and provides good parameter values for the mathematical model.

The following figure shows a plot of the fraction of infected machines (according to our model) at each of three quality ratings as a function of time. The plot was generated for one subgroup containing 100 machines at rating 1, 100 at rating 0.5, and 200 at rating 0.25, with a timeout value of 15. We used $delta = 0.01$, and set each machine's infectivity proportional to its rating. We applied the above algorithm to determine susceptibilities that would give the desired fraction of infected machines at each rating within timeout rounds. We then ran the recurrence relations with these computed parameters to produce the figure.

In the figure, the vertical line indicates the timeout value. From left to right, the other three lines indicate the fraction of infected machines at rating 1, 0.5, and 0.25 respectively. As we can see, by round 15, we do have 1/2 of the rating 0.5 machines infected, and 1/4 of the rating 0.25 machines. Recall that we used $\delta = 0.01$. At round 15, 99 percent of the rating 1 machines are infected. Thus, the computed parameters give us the results we expected.

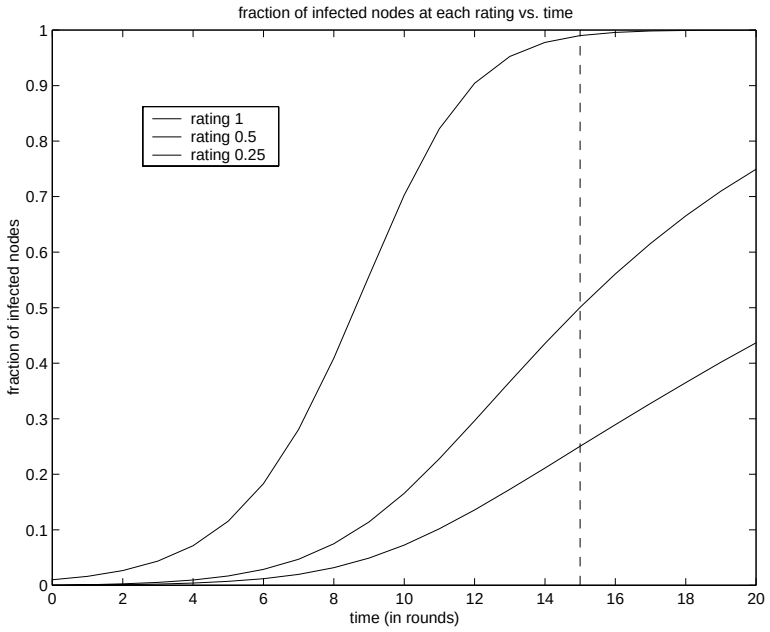

Figure 1: Plot of the fraction of infected machines vs. number of gossip rounds, using gossip parameters predicted by our model. Parameters were predicted for one subgroup with 100 machines at rating 1, 100 at rating 0.5, and 200 at rating 0.25, with a timeout value of 15.

5 Results

We now return to the problem of estimating gossip overhead when individual machines may belong to multiple subgroups at varying quality ratings. Each of the subgroups gossip independently, and for the sake of our analysis, we assume that all of the data to be gossiped within a subgroup fits inside one gossip message. By gossip overhead, we mean the expected number of gossip messages initiated and received by a machine in a gossip round, for all of its subgroups.

We are interested in a variety of possible subgroup configurations, but for now restrict our attention to the case where each machine belongs to the same number of subgroups at each rating, and where each subgroup has the same number of participating machines at each rating and the same timeout value. This has the advantage that all machines will have the same expected gossip overhead. We then use the parameter prediction method from the previous section to determine what this overhead will be.

Recall that in our gossip protocol, a subgroup member m gossips to member n with probability $I_m S_n$, where I_m is the infectivity of m and S_n is the susceptibility of n. Since we are now considering multiple subgroups, and a machine can have different ratings within different subgroups, we will revise our notation to $I_g(m)$, $S_g(n)$ for the infectivity of m within subgroup g, and the susceptibility of n in g, respectively. Hence the expected number of messages m will sent to n in a round as a member of g will also be $I_g(m)S_g(n)$.

Each subgroup gossips independently, so the total number of gossips initiated by a machine m will be the sum of the number it initiated as a member of each subgroup. By linearity of expectation, the expected number of messages sent by m as part of g will be $\sum_{n \in g} I_g(m)S_g(n)$, and the number overall will be:

$$\sum_{g : m \in g} I_g(m)(\sum_{n \in g} S_g(n))$$

Note that when all machines belong to the same number of subgroups at each rating, and all subgroups have the same number of members at each rating, $\sum_{n \in g} S_g(n)$ will be a constant independent of g, and $\sum_{g:m \in g} I_g(m)$ will be a constant independent of m. Hence all machines will have the same outgoing gossip overhead.

Similarly, the expected number of messages m will receive in a gossip round is:

$$\sum_{g:m \in g} S_g(m)(\sum_{n \in g} I_g(n))$$

which will also be independent of m.

Hence, given the infectivities and susceptibilities determined by our algorithm for each subgroup, we can compute the expected total gossip overhead for each machine.

To better understand the effect of quality ratings, timeout values, and subgroup size on gossip overhead, we compute the expected gossip overhead as a function of timeout value and subgroup size for two different subgroup configurations. The results are shown in the figure below. In one configuration, each machine is in 2 subgroups, with rating 1 for both subgroups. In the second configuration, each machine is in 4 subgroups, 1 at rating 1, 1 at rating 0.5, and 2 at rating 0.25. Note that both of these ratings that sum to 2.

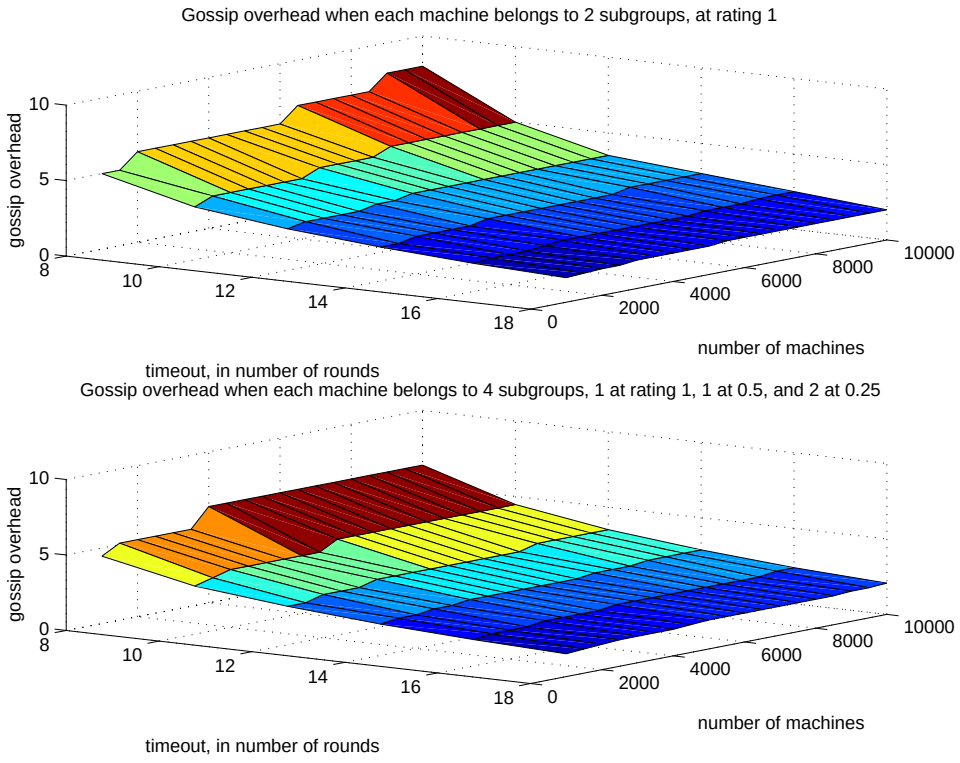

Figure 2: Comparison of gossip overhead as function of timeout and subgroup size for two different subgroup configurations. In one case, each machine belongs to two subgroups at rating=1. In the other, each machine belongs to 4 subgroups, at ratings 1, 0.5, 0.25,and 0.25.

As we can see from the figure, overhead generally increases with decreasing timeout values. Decreasing the timeout value can cause a large increase in gossip overhead when the timeout is small. Also, increasing the group size causes only a very small increase in gossip overhead for fixed timeout value when the timeout

value is moderate. For such timeout values, both configurations give a gossip overhead of around 2 for subgroup sizes up to 10,000.

One interesting feature is that the configuration with 4 subgroups has gossip overhead comparable to, and sometimes better than, the configuration with only 2 subgroups. This suggests that having variable quality ratings for subgroup members can be a useful way to allow a machine to belong to many subgroups while maintaining low gossip overhead.

6 Conclusions and Future Work

We presented the gravitational gossip protocol and showed that it can be used for reliable subgroup communication. It allows processes to be members of several subgroups at reduced quality ratings with comparable overhead to that of machines in a few subgroups at higher quality ratings. Furthermore, it defines a clear tradeoff between information quality and gossip overhead which can be chosen by each member individually.

In the interests of brevity and deadlines, there are several topics that were left out of this writeup.

• The data presented above is based on a mathematical model. It remains to be demonstrated how well this corresponds to reality. There are implementation details that were ignored in this writeup.

• Ken Hopkinson is currently implementing a real application of gravitational gossip techniques to a problem in the electric power grid.

• We only considered very restricted subgroup configurations. We would like to analyze the protocol overhead with more realistic subgroups.

• We only considered the case in which subgroups gossip independently. If there is considerable overlap between subgroups, then they could often share gossip messages. We are looking at ways to share gossip messages between subgroups, in order to generate fewer messages but have each message contain more data samples.

• In this paper, we only considered gossip within a single LAN. We are interested in analyzing gossip subgroup communication in more general network topologies.

7 Acknowledgements

We would like to thank Carlos Castillo-Chavez for enlightening discussions about epidemic modelling.

References

[1] N. T. J. Bailey. *The Mathematical Theory of Infectious Diseases and its Applications (second edition).* Hafner Press, 1975.

[2] Carlos Castillo-Chavez. Personal communication. Department of Biometrics, Cornell University.

[3] A. Demers et al. Epidemic algorithms for replicated database maintenance. In *Proceedings of 6th ACM Symposium on Principles of Distributed Computing*, pages 1–12, Vancouver, British Columbia, 1987.

[4] K. Birman et al. Bimodal multicast. Technical Report Department of Computer Science Technical Report TR-98-1665, Cornell University, Ithaca, NY, 1998.

[5] L. Rodrigues et al. A dynamic light-weight group service. In *Proceedings of the 15th IEEE Symposium on Reliable Distributed Systems*, Niagara on the Lake, Canada, 1996.

[6] K. Guo. Scalable membership detection protocols. Technical Report Department of Computer Science Technical Report TR-98-1684, Cornell University, 1998.

[7] K. Guo and L. Rodrigues. Dynamic light-weight groups. In *Proceedings of the 17th IEEE International Conference on Distributed Computing Systems*, Baltimore, MD, 1996.

[8] B. Pittel. On spreading a rumor. *SIAM Journal of Applied Mathematics*, 47:213–223, 1987.

[9] R. van Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. In *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware '98)*, pages 55 – 70, The Lake District, England, 1998.