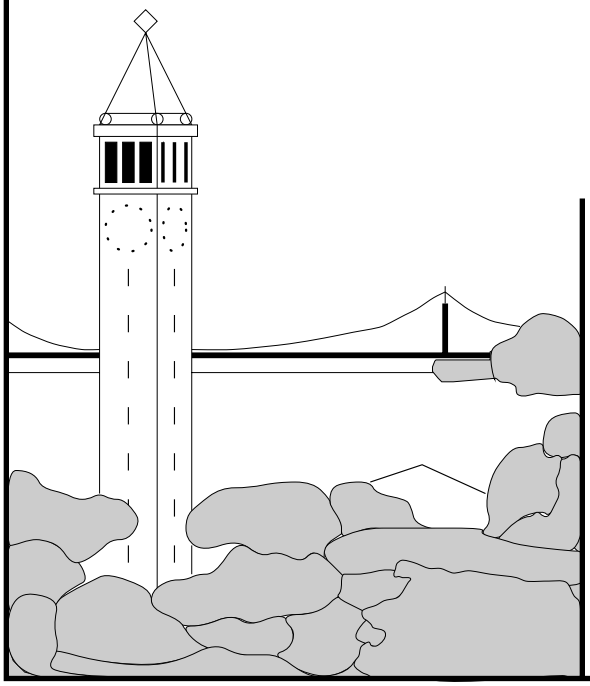
Technical Report: Randomized Rumor Spreading with Fewer Phone Calls*

Kirsten Hildrum *Sean Ma* *Satish Rao*
University of California, Berkeley
{hildrum@cs,seanma@hkn.eecs,satishr@cs}.berkeley.edu



Report No. UCB//CSD-04-1329

June 2004

Computer Science Division (EECS)
University of California
Berkeley, California 94720

* Kirsten Hildrum supported by NSF career award ANI-9985250, NFS ITR award CCR-0085899, and California MICRO award 00-049. Satish Rao supported in part by NSF grant CCR-0105533

Abstract

Rumor spreading algorithms are a useful way to disseminate information to a group of players in the presence of faults. Rumors are either spread by pushing, in which the players knowing the rumor call other players at random and spread the rumor, or by pulling, where players who do not know the rumor call other players and ask for any new rumors.

The efficiency of the algorithms is often measured in the number of transmissions (the number of times the rumor is shared), but could also be measured in phone calls (the number of connections that need to be made between players.)

In this technical report, we make two observations to reduce the number of phone calls. The first idea spreads the rumor in a randomized way and then uses a fixed-connection network (such as a tree) to finish sharing. This uses many fewer phone calls than pure rumor spreading, though some nodes may not be notified. Second, we observe that using both push and pull but pulling infrequently can reduce the number of phone calls if rumors enter the network at a slow rate.

1 Introduction

Suppose a node has an update that it would like to send to $n-1$ other nodes. The node with the update could send the message to every other node directly, but this takes $\theta(n)$ steps. Another option is to build a fixed-connection network (e.g., a phone tree). Then, a node with an update sends the update to its parent and its children, and the nodes that receive it send it on in a similar way. The update reaches everyone in $O(\log n)$ steps, and each node—including the node initiating the rumor—only does constant work. Further, the total number of messages and the total number of phone calls is optimal, $n-1$. But if one node in the tree fails, a large subtree becomes unreachable.

One solution to this problem is the use of rumor spreading algorithms. These algorithms have two basic operations: *push* and *pull*. In a push, a node that has the update and randomly sends the rumor to another participant. In a pull, a node without the rumor asks a random node for any new rumors.

We consider two measures of efficiency for rumor spreading algorithms. The first is the number of connections (or phone calls) made. The second measure is the number of times the rumor is transmitted. If two nodes connect but neither of them know the rumor, a phone call may not result in a transmission. If the cost of the phone calls is amortized over many rumors, it makes sense to count transmissions rather than phone calls, but if rumors are rare, the number of phone calls may be more important. (Also important is the number of rounds, or steps, but we do not consider that here.)

Among the first to use epidemic algorithms is Demers et. al. [DGH$^+$87], in the context of maintaining loose consistency among replicated databases. Since then, many research have improved on various properties of these algorithms. See, for example, [AH96], which relaxes the assumption that all nodes know the other nodes, or [CK02, EGH$^+$03, GP96, MMR99, Kel99, MS03] which are more robust to faults. Rabinovich, Gehani and Kononov [RGK96] makes determining whether a transmission is needed more efficient. Rumor spreading protocols have been used to maintain data structures in Peer-to-Peer systems [RGRK, RGRK03, ACMD$^+$03, DHA03], and locate objects in networks [KKD01]. Kempe and Kleinberg presented some impossibility results in [KK02]. Broadcasting itself (without necessarily rumor spreading) is a well-studied problem, see for example [HKMP96, HHL88] for a survey of techniques. The effects of faults on broadcasting has also been well studied; see for example, [LMS98], which examines the fault-tolerance of fixed connection networks.

The observations here draw most from two papers. Karp et al. [KSSV00] takes a theoretical approach and give an algorithm that distributes a rumor in $O(\log n)$ rounds, transmitting the message $O(n \log\log n)$ times, and uses $O(n \log n)$ phone calls. Birman et al. [BHO$^+$99] uses epidemic propagation in addition to a fixed connection network for a more reliable multicast.

First, we suggest that using a little bit of rumor spreading with a fixed connection network can give some fault-tolerance benefits of randomize rumor spreading with the efficiency of a fixed-connection network. Note that in a fixed connection network, the number of transmissions is $O(n)$. Birman et al. [BHO$^+$99] also combine rumor spreading with a fixed connection network to make a more reliable multicast.

Second, we describe a minor modification of Karp et. al's algorithm that reduce the rate of the pull steps so as to use only $\theta(n \log\log n)$ *phones calls* as compared to their $\theta(n \log n)$. Datta, Hauswirth, and Aberer [DHA03] do something similar, using pull only when nodes wake up or have not heard news recently.

Rumor spreading protocols have found applications to many areas, and many variations of them have been considered. For example, Kempe, Kleinberg, and Demers in [KKD01] introduced the notion of spa-

% failed	tree	push	push+tree
0	100	88	100
10	2	63	88
20	0	30	60
30	0	10	27
40	0	3	8
50	0	1	1

Figure 1: The failure rate of a tree alone, push for the same number of rounds, and the combination of the two. The simulation used 1048576 nodes and the results are averaged over ten runs.

	phone calls		transmissions
n	push+tree	push&pull	push&pull
16384	2.2	15.3	5.2
64436	2.1	17.4	5.9
262144	2.0	19.6	6.3
1048576	2.0	22.0	6.7
4194304	2.0	23.4	7.0

Figure 2: The number of messages sent for push+tree and push&pull. There are n nodes in the network, and 10% are dead. The push&pull alogrithm is run until all nodes know the rumor. (Neither push nor pull alone performed as well as push & pull.)

tial gossip, in which nearby nodes are more likely to contacted than distant nodes.

The Karp et. al. algorithm proceeds by pushing and pulling messages for $\log n \pm O(\log \log n)$ steps. In the beginning, the process essentially doubles the number of players who know the message. (Pushing and pulling triples the number of messages at each step.) This geometric increase means that up until this point, only $O(n)$ transmissions had occurred. However, $n \log n$ phone calls have been made, due to the pulling. Most of the pull connections are between nodes that do not have the rumor, so the count of transmissions is unaffected by this.

Both our extensions make use of what we will call a push phase–that is, $\log_2 n$ rounds of simple push communication. In this phase, a rumor starts with a birthday of zero. With each round, every node with the rumor pushes it to a random node, and increases the age of the rumor by one. Once the age is $\log_2 n$, we stop. Up until this time, there are no pull requests, so only $O(n)$ phone calls and transmissions have been made.

2 Push+tree

First we perform the push phase described above. This gets around many of the faults in the fixed connection network. Second, we perform a second phase using a fixed connection network. The push phase and the fixed-connection network each use only a linear number of messages, so the total number of messages is linear. We use this technique with two networks. The first is a tree, chosen because it is very simple to construct and update, and very vulnerable to faults, and so may be the worst case. The second is a low-degree random graph.

To test the usefulness of the push+tree scheme, we used simulation. Our simulations used a tree that was as close to balanced as possible. We present two results. First, Figure 1 shows how the failure rate of the networks affects the number of nodes left informed by the push+tree scheme of Section 2. With the tree alone, even a few failures in the network prevent the message from reaching most of the nodes. In contrast, an initial push phase significantly increases the percentage of nodes that receive the message, and using a tree is more effective than adding additional rounds of push communication.

% failed	random	random+push
0	100	100
10	100	100
20	98	99
30	92	93
40	42	72
50	0	15

Figure 3: Success rate of message dissemination using a random degree-three graph.

In Figure 2, we compare the number of phone calls and the number of transmissions to those of the push+tree scheme. (Recall that a phone calls occurs whenever two nodes connect; a transmission occurs when at least one of them knows the rumor.) Note that for the push+tree scheme, the number of transmissions and the number of phone calls is essentially equal.

As we noted above, a tree is a ineffective structure for dissemination in a faulty environment, since one failure disconnects a tree. For comparison, we built a low-degree random graph and ran the same set of tests. The random graph in this case is the union of three random matchings. The simulation results in Figure 3 show that at low failure rates, most nodes in the random graph are notified, even without the push phase. However, at the higher failure rates, the push scheme does help some.

3 A slower pull

Once at least half the nodes are informed, pull communication requires very few rounds and very few messages to reach the rest of the nodes, since it roughly squares the fraction of uninformed nodes at each step [DGH+87,KSSV00]. Once half the nodes in the network know the rumor, only $O(\log \log n)$ rounds of pulling are needed before all the rest know.

But during the $O(\log n)$ rounds to inform that half, every node makes a pull phone call. Most of those phone calls are wasted, since neither node knows the rumor, and no transmission can occur. In this section, we seek to limit these wasted phone calls.

If we had a global counter measuring the age of the rumor, we could have nodes with the rumor push for $\log_2 n$ steps, and then cease pushing. The nodes that do not have the rumor then start to pull, and pull for $O(\log \log n)$ steps. This would result in $O(n \log \log n)$ phone calls, which is optimal. However, this makes the unreasonable assumption that all nodes know when to start pulling. We make the following observation:

Observation 1 *Push and pull need not happen at the same rate. If rumors are pushed for* $\log_2 n$ *steps, while pulling occurs every* $O(\log n / \log \log n)$ *steps, then rumors will arrive at all the nodes in* $O(\log n)$ *steps with* $O(n \log \log n)$ *phone calls.*

Thus, a two-phase algorithm can reduce the number of wasted phone calls. The first phase is a push phase lasting $\log_2 n$ steps. (The rumor comes with a time-to-live of $\log_2 n$ that decreases by one at each step. Once the time-to-live is zero, nodes stop pushing it.)

Simultaneously, each node pulls once every $O(\log n / \log \log n)$ time steps. Between two pulls by the same node, every other node will have performed one pull, so this acts as a slowed-down version of the pull phase described above. Emulating one pull phase of the above paragraph takes time $O(\log \log n)$ pulls, or $O(\log n/(\log \log n)) \cdot \log \log n = O(\log n)$ rounds. A rumor spends $\log_2 n$ rounds being pushed (and is transmitted $O(n)$ times during that phase), and then, and another $O(\log_2 n)$ being rounds being pulled to the remaining nodes, but the total number of connections needed is only $O(n \log \log n)$.

Wasted phone calls and the rumor rate Whether this technique saves phone calls or not depends on how many rumors are in the network. Regardless of how many rumors are in the network, each node still pulls once every $O(\log n / \log \log n)$ steps. As with the original scheme, if the number rumors is very low, the pull phone calls occur when there is no rumor currently spreading in the network, and so are wasted. For example, if there are no rumors in the network, every pull phone call is wasted. But if rumors arrive approximate $O(\log n)$ steps, and pulls occur approximately $O(\log n / \log \log n)$ steps, then most pulls occur at a time when there is a rumor currently spreading in the network. The disadvantage of the slower pull is that it increases the time until all players know the rumor.

On the other hand, if the rumor rate is high, then even pulling at every step results in the transmission of some rumor, and so there are no "wasted" phone calls, making the slower pull unnecessary. Thus, when many rumors are spreading, the slower pull described gives no benefits while delaying the time at which all nodes know the rumor.

To minimize the number of wasted phone calls, the pull rate should depend on the rumor rate. When there are many rumors in the network, the pulls should be frequent. When rumors are rare, the pull rate should be slowed to match.

References

[ACMD+03] Karl Aberer, Philippe Cudré-Mauroux, Anwitaman Datta, Zoran Despotovic, Manfred Hauswirth, Magdalena Punceva, and Roman Schmidt. P-grid: a self-organizing structured p2p system. *SIGMOD Rec.*, 32(3):29–33, 2003.

[AH96] James Aspnes and William Hurwood. Spreading rumors rapidly despite an adversary. In *Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing*, pages 143–151. ACM Press, 1996. One caller per step.

[BHO+99] Kenneth P. Birman, Mark Hayden, Oznur Ozkasap, Zhen Xiao, Mihai Budiu, and Yaron Minsky. Bimodal multicast. *ACM Trans. Comput. Syst.*, 17(2):41–88, 1999.

[CK02] Bogdan S. Chlebus and Dariusz R. Kowalski. Gossiping to reach consensus. In *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, pages 220–229, 2002.

[DGH+87] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, and John Larson. Epidemic algorithms for replicated

database maintenance. In *Proc. of the sixth annual ACM Symp. on Principles of distributed computing*, pages 1–12, 1987.

[DHA03] Anwitaman Datta, Manfred Hauswirth, and Karl Aberer. Updates in highly unreliable, replicated peer-to-peer systems. In *Proceedings of the 23rd International Conference on Distributed Computing Systems*, page 76. IEEE Computer Society, 2003.

[EGH+03] P. Th. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec. Lightweight probabilistic broadcast. *ACM Trans. Comput. Syst.*, 21(4):341–374, 2003.

[GP96] L. Gasieniec and A. Pelc. Adaptive broadcasting with faulty nodes. *Parallel Computing*, 22:903–912, 1996.

[HHL88] S.M. Hedetniemi, S.T. Hedetniemi, and A.L. Liestman. Survey of gossiping and communication networks. *Networks*, 18(4):319–349, 1988.

[HKMP96] J. Hromkovic, R. Klasing, B. Monien, and R. Peine. *Dissemination of Information in Interconnection Networks (Broadcasting & Gossiping)*. Kluwer Academic Publishers, 1996.

[Kel99] Peter J. Keleher. Decentralized replicated-object protocols. In *Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*, pages 143–151, 1999.

[KK02] David Kempe and Jon Kleinberg. Protocols and impossibility results for gossip-based communication mechanisms. In *IEEE Symposium on Foundations of Computer Science*, pages 471–480, 2002.

[KKD01] David Kempe, Jon Kleinberg, and Alan Demers. Spatial gossip and resource location protocols. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 163–172, 2001.

[KSSV00] Richard M. Karp, Christian Schindelhauer, Scott Shenker, and Berthold Vöcking. Randomized rumor spreading. In *IEEE Symp. on Foundations of Computer Science*, pages 565–574, 2000.

[LMS98] F. Thomson Leighton, Bruce M. Maggs, and Ramesh K. Sitaraman. On the fault tolerance of some popular bounded-degree networks. *SIAM Journal on Computing*, 27(5):1303–1333, 1998.

[MMR99] Dahlia Malkhi, Yishay Mansour, and Michael K. Reiter. On diffusing updates in a byzantine environment. In *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, page 134. IEEE Computer Society, 1999.

[MS03] Yaron M. Minsky and Fred B. Schneider. Tolerating malicious gossip. *Distributed Computing*, 16(1):49–68, 2003.

[RGK96] Michael Rabinovich, Narain H. Gehani, and Alex Kononov. Scalable update propagation in epidemic replicated databases. In *Proceedings of the 5th International Conference on Extending Database Technology*, pages 207–222, 1996.

[RGRK] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. Handling churn in a DHT. To appear in USENIX '04 Annual Technical Conference.

[RGRK03] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. Handling churn in a DHT. Technical Report UCB//CSD-03-1299, UC Berkeley, 2003.