

## 1 Analysis of Push-Pull Gossip

Last class, we started analyzing the time and message requirements of a Push-Pull based gossip algorithm for disseminating a single message [1]. We showed that after a short startup phase of  $O(\log \log n)$  rounds, at least  $\log^4 n$  nodes have the message with high probability. After that point, the number of nodes will be enough to apply Chernoff Bounds to make sure that the process behaves very close to its expectation.

**Exponential Growth** : Next, we looked at an *exponential growth phase*, which was defined to last from the time when  $\log^4 n$  nodes have the message until  $n/\log n$  nodes do. For this phase, we let  $s_t$  denote the number of informed nodes after  $t$  time steps, and  $m_t$  the number of messages containing the rumor that were sent in round  $t$ . Because each node calls out, and is called, once in expectation, we had that  $E[m_t] = 2s_{t-1}$ , and that  $m_t$  is sharply concentrated around its expectation by Chernoff Bounds. So  $m_t = (2 \pm o(1/\log n))$  with high probability.

Some of these  $m_t$  messages will not succeed in informing new nodes. The reasons could be two-fold: either they reach an already informed node, or multiple messages reach the same uninformed nodes. The probability that a given message reaches an already informed node is  $s_{t-1}/n$ , and the probability of a collision with another message is  $m_t/n$ . Hence, the expected number of informed nodes after  $t$  iterations is

$$\begin{aligned} E[s_t] &= s_{t-1} + m_t(1 - m_t/n - s_{t-1}/n) \\ &= s_{t-1}(1 + (2 \pm o(1/\log n))(1 - O(1/\log n))) \\ &= s_{t-1}(3 \pm O(1/\log n)). \end{aligned}$$

In the second step here, we used the fact that  $s_{t-1} \leq n/\log n$ . Again, we can write the number of successful messages (those that inform a new node) as a sum of 0-1 random variables, and apply a Chernoff Bound, showing that the value of  $s_t$  will be sharply concentrated (to within a multiplicative  $1 \pm 1/\log n$ ) around its expectation. When all iterations have outcomes close to the expectation (which happens with high probability by a Union Bound over all these iterations), we reach  $n/\log n$  nodes in  $\log_3 n + O(\log \log n)$  rounds.

**Quadratic Shrinking** : Once  $n/\log n$  nodes are reached, each uninformed node has sufficiently high probability of reaching an informed node in its Pull attempt, so we will ignore Push transmissions, and only use Pull in the analysis. We write  $u_t$  for the number of uninformed nodes after  $t$  iterations. Then, for any node that is uninformed in iteration  $t - 1$  and makes a uniformly random Pull attempt, the probability that it stays uninformed is  $u_{t-1}/n$ . As a result, the expected fraction of uninformed nodes after the  $t^{\text{th}}$  iteration is  $E[u_t/n] \leq (u_{t-1}/n)^2$ , leading to quadratic shrinking. Again, the calls for different nodes are independent, so we can write  $u_t$  as a sum of independent 0-1 random variables, and so long as the number of uninformed nodes  $u_{t-1}$  is large enough ( $u_{t-1} \geq \sqrt{n} \log^4 n$ ), quadratic shrinking will occur with high probability by Chernoff Bounds.

Because the number of uninformed nodes thus shrinks doubly exponentially, within  $O(\log \log n)$  rounds, the number of remaining uninformed nodes is at most  $\sqrt{n} \log^4 n$ , with high probability.

**Finishing** : Finally, once all but  $\sqrt{n} \log^4 n$  of the nodes are informed, each node has probability  $1 - \frac{\log^4 n}{\sqrt{n}}$  of being informed during any one Pull call it makes, and successive rounds are independent. Hence, after three rounds, the probability of any one node not being informed is at most  $\frac{\log^{12} n}{n^{3/2}}$ , and by a Union Bound, all nodes are informed after the additional three rounds with probability at least  $\frac{\log^{16} n}{n}$ .

Taking all this together, we have proved that with high probability, all nodes will learn the message within  $\log_3 n + O(\log \log n)$  rounds. To analyze the total number of messages sent, we notice that the startup, quadratic shrinking, and finishing phases take only  $O(\log \log n)$  rounds total, so the total number of messages sent in those rounds is  $O(n \log \log n)$ . During the exponential growth phase, at most  $n/\log n$  nodes are active, so no more than  $n/\log n$  messages are sent each round, thus no more than  $O(n)$  total. As a result, the total message complexity is  $O(n \log \log n)$ .

We should notice that the protocol, as presented here, is very vulnerable to faults. It needs to be run for exactly  $\log_3 n + O(\log \log n)$  rounds. If the number of rounds is  $\Omega((1 + \epsilon) \log_3 n)$ , then  $\Omega(\epsilon n \log n)$  messages are sent during those extra  $\epsilon \log_3 n$  rounds, violating the goal of  $O(n \log \log n)$  messages. Conversely, if the number of rounds is  $O((1 - \epsilon) \log_3 n)$ , then with high probability, some nodes will remain uninformed.

Karp et al. [1] show how a more sophisticated “Median Counter” algorithm achieves the same kind of guarantee while being more resilient to failures.

A second question is how low we can push the number of messages sent. Of course, if we do away with gossip, and build a tree instead, then  $n - 1$  messages suffice to inform all nodes. Even using uniform gossip, we can get away with  $n - 1$  messages if we are willing to sacrifice the completion time. A protocol achieving this would be one where the initial owner of the message is the only one to transmit it, and does so whenever he calls (or is called by) an uninformed node. This turns the problem into a *coupon collector* problem: the message has reached everyone if everyone has called (or been called by) the initial message holder. By well-known results, this takes  $\Theta(n \log n)$  rounds with high probability.

Obviously, this type of protocol goes completely against the idea of using the network in spreading a message. Karp et al. show that under “reasonable” restrictions on the protocol,  $\omega(n)$  messages are necessary. The first result concerns *address-oblivious* protocols: in each round, a node must decide whether to transmit to its communication partner(s) without knowing their identity. In particular, nodes do not know if the communication partner already has the message. Notice that the Push-Pull protocol we analyzed above is address-oblivious.

**Theorem 1 ([1])** *Any address-oblivious protocol has to send at least  $\Omega(n \log \log n)$  messages to inform all nodes.*

Even without the restriction, lower bounds can be proved, trading off the completion time and message complexity:

**Theorem 2 ([1])** *Any protocol reaching all but  $O(1)$  of the nodes in  $O(\log n)$  rounds, and using uniform gossip, must send at least  $\omega(n)$  messages.*

An interesting question along the lines of reducing the number of messages sent is how many nodes are informed, and how many rounds it takes, when each node forwards the message  $k$  times after receiving it, for some constant  $k$  (such as  $k = 2$ ).

## 2 Averaging using Gossip

So far, we have focused on gossip only as a means to disseminate a single message to all nodes. In reality, this is a simplistic abstraction. When each node has a message that is to be shared with all other nodes, then we could consider a protocol in which each node always forwards all messages it holds. In that case, we are essentially “super-imposing” the simple gossip protocol for all source messages, so all nodes will learn all messages in  $O(\log n)$  rounds with high probability. Unluckily, the message complexity will be  $\Omega(n^2 \log \log n)$ .

However, if we assume that messages are atomic, i.e., cannot be aggregated, then there is not much beyond this bound that can be done — even dissemination along a tree will take  $\Omega(n^2)$  messages total.

We may be able to send significantly less data around if the messages can be aggregated. Instead of forwarding each message individually, we could then perform aggregation within the network. For instance, assume that each node  $i$  just holds one number  $x_i$  (the number of files, or available memory, or a temperature measurement), and we want to compute the average  $\bar{x} = \frac{1}{n} \sum_i x_i$  of these numbers. Using a tree, values could be added up (and nodes counted) on the way to the root; the root could then compute the average, and distribute it back to all nodes using the tree. Using a tree is not a very fault-tolerant approach, so we want to use gossip instead.

Our gossip-based algorithm is very simple. Each node  $i$  maintains only two values, a *sum*  $s_i$ , and a *weight*  $w_i$ . The sum is initialized to  $s_i := x_i$ , and the weight to  $w_i := 1$ . Then, in each round, each node executes the following protocol Push-Sum [2]:

---

**Algorithm 1** Push-Sum

---

- 1: Send the pair  $(s_i/2, w_i/2)$  to yourself and a uniformly randomly chosen other node.
  - 2: Let  $J_i$  be the set of all nodes that have sent a message to  $i$  in this round.
  - 3: The new values are  $s'_i := s_i/2 + \sum_{j \in J_i} s_j/2$  and  $w'_i := w_i/2 + \sum_{j \in J_i} w_j/2$ .
  - 4: Keep track of  $s_i/w_i$  as approximation of the average.
- 

One useful fact that we can notice right away about this protocol is its property of *mass conservation*: at any point of the execution, we always have that  $\sum_i s_i = \sum_i x_i$ , and  $\sum_i w_i = n$ . The sums and weights get redistributed, but not truly changed. Next class, we will prove the following theorem about the convergence of Push-Sum to the true average.

**Theorem 3** *If all  $x_i$  are non-negative, then within  $O(\log n + \log \frac{1}{\delta} + \log \frac{1}{\epsilon})$  rounds, all estimates  $s_i/w_i$  are within  $(1 \pm \epsilon)$  of the true average  $\bar{x} = \frac{1}{n} \sum_i x_i$ , with probability at least  $1 - \delta$ .*

Thus, the error drops exponentially in the number of iterations we run.

## References

- [1] R. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking. Randomized rumor spreading. In *Proc. 41st IEEE Symp. on Foundations of Computer Science*, pages 565–574, 2000.
- [2] D. Kempe, A. Dobra, and J. Gehrke. Computing aggregate information using gossip. In *Proc. 44th IEEE Symp. on Foundations of Computer Science*, pages 482–491, 2003.