CS 595: Sensor Network Seminar **Fall 2003**

Lecture 12 — December 9, 2003

Presenter: Nahur Fonseca *BOSTON UNIVERSITY* *Scribe: Dhiman Barman*

In today's student's presentation, we learned Gossip-style protocols for doing aggregation in the context of sensor networks [4].

12.1 Gossip Problem

Over the last-decade, we have seen a revolution in connectivity between computers, and a resulting paradigm shift from centralized to highly distributed systems. With massive scale also comes massive instability, as node and link failures become the norm rather than the exception. For such volatile systems, decentralized gossip-based protocols are emerging as an approach to maintaining simplicity and scalability while achieving fault-tolerant information dissemination. In gossip-based protocols, each node contacts one or a few nodes in each round, and exchanges information with these nodes. Gossip-based protocols usually do not required error recovery mechanisms, and thus enjoy a large advantage in simplicity, while often incurring only moderate overhead compared to optimal deterministic protocols, such as the construction of data dissemination trees.

The gossip problem can be realized as, *There are n ladies, and each of them knows some item of gossip not known to the others. They communicate by telephone, and whenever one lady calls another, they tell each other all that they know at that time.* The question is *how many calls are required before each gossip knows everything*?

To derive the number of calls needed, we consider a recurrence relation. Let $f(n)$ be the minimum number of calls. Clearly, $f(1) = 0, f(2) = 1, f(3) = 3$ and $f(4) = 4$. For $n > 4, 2n - 4$ calls are necessary and sufficient. Baker and Shostak [1] have analyzed a gossip based protocol showing the above proof. We can easily derive the result by solving the recursion as

$$\begin{aligned} f(n) &= f(n-1) + 2 && (12.1) \\ &= 2(n-4) + 4 = 2n - 4 && (12.2) \end{aligned}$$

In the remainder of the scribe we are going to study how to compute aggregates using gossiping, specifically gossip-style protocols under practical considerations. We will study uniform gossiping and its generalizations.

12.1.1 General Communication Model

In general communication model of gossiping, we will consider that communication nodes form a complete graph and the edges of the graph are full-duplex links. There is no limit on the message size that can be transmitted to gossips. Also, there is no randomness. Several different proofs have been suggested with more restrictions on the graph model.

12.1.2 Models with Restriction

Harary and Schwenk [3] showed that the number of calls needed in gossiping over trees is $f(n) = 2n - 3, n \geq 2$. For the connected graph the condition is $2n - 4 \leq f(n) \leq 2n - 3$ for a strongly connected graph the condition is $f(n) = 2n-2$. It is to be noted that some of the graphs are more representative models of communication model of sensor network.

Time bounds for convergence, $t(n)$ can also be given for special graph structures. Bevalas [2] showed that for n even, $t(n) = \lceil \log n \rceil$ and for n odd, $t(n) = \lceil \log n \rceil) + 1$. Recently, Labahn [5] has determined the minimum number of edges in graphs that support minimum time gossiping for some even values of n. For hypergraphs, it has been shown that $t(n,k) = \lceil \log_k n \rceil$ if $k|n$ and $t(n,k) = \lceil \log_k (n/(k-1)) \rceil + 1$ otherwise.

12.2 Computing Aggregates with Gossiping

Over the last decade, we have seen a revolution in connectivity between computers, and a resulting paradigm shift from centralized computation to highly distributed systems. For example, large-scale peer-to-peer (P2P) networks with millions of servers are being used or designed for distributed information storage and retrieval, and advances in hardware are leading to the augmentation of our physical environment with sensor networks consisting of hundreds of thousands of small sensor nodes. Applications for such large-scale distributed systems have three salient features: First, the dynamics of large-scale distributed systems are very different due to adversarial environment, lossy network and failure prone. Second, due to the large number of nodes and the volatility of the system, any reliance on central coordination will limit the system's scalability. Third, due to the large scale of the system, the values of the aggregate functions over the data in the whole network are often more important than individual data at nodes. For example, in a sensor network with temperature sensors, we are often more interested in the average or median temperatures by all sensors in an area rather than the single measurement at an individual sensor.

12.2.1 Motivation behind Gossiping

Gossip-style protocols have very fast convergence complexity and is given by $\mathcal{O}(\log n)$. Comparing to Distributed Hash Table which is also fast has complexity $\mathcal{O}(\sqrt{n})$ in sensor networks. There is no need for special treatment to assure reliability. It is assumed that the message sizes will be small given by $\mathcal{O}(\log n + |x_{largest}|)$ bits.

12.2.2 Tradeoffs

Although gossiping seems appropriate for large-scale distributed network but there are tradeoffs in gossip-style protocols. Gossiping generates a lot of messages , $O(n \log n)$. There are issues about *eventual consistency* as data may change before gossiping is complete. Poorly aggregate functions can cause a lot energy consumption. Moreover, it is hard to know the terminating conditions but gossip-style protocols do not need any synchronization.

12.2.3 Algorithms

We describe the generalized "push" gossip protocol in Algorithm 1.

Algorithm 1 Generalized push gossip protocol

```
1: Take a graph
2: At each time step, select a non-negative share $\alpha_{t,i,j}$ for each node $j$ such that sum is 1.
3: Send a share of node's $i$ info to every node $j$
```

To compute aggregate function "sum", we describe the algorithm in Algorithm 2.

Algorithm 2 Push-sum

```
1: At time 0, the pair $(s_{0,i} := x_i, w_{0,i} := 1)$ is sent to $i$ itself
2: For the subsequent time steps, do the following:
3: Let $\{(\hat{s}_r, \hat{w}_r)\}$ be all pairs sent to $i$ in round $i - 1$
4: Let $s_{t,i} := \sum_r \hat{s}_r, w_{t,i} := \sum_r \hat{w}_r$
5: Choose a target $f_r(t)$ uniformly at random
6: Send the pair $(1/2s_{t,i}, 1/2w_{t,i})$ to $f_t(i)$ and $i$ (yourself)
7: $s_{t,i}/w_{t,i}$ is the estimate of the average in step $t$.
```

If we want to compute the sum instead of the average, then we only need to apply a small change: instead of all nodes starting with weight $w_{0,i} = 1$, only one node starts with weight 1, while all others start with weight 0. We then obtain exactly the same kind of approximation schemes.

We define *diffusion speed* as a notion which characterizes how quickly values originating with multiple sources diffuse evenly through a network, for a given communication mechanism.

To track the diffusion of a node i's value under a given communication mechanism, we define the following vector-based version of the protocol. Each node i locally maintains a n-dimensional contribution vector $v_{t,i}$. Initially, it sends the vector e_i to itself. In all subsequent rounds, the protocol is:

Algorithm 3 Push-vector

```
1: Let $\{\hat{v}_r\}$ be all vectors sent to $i$ in round $t - 1$
2: Let $v_{r,i} := \sum_r \hat{v}_r$
3: Choose shares $\alpha_{t,i,j}$ for all nodes $j$
4: Send $\alpha_{t,i,j} \cdot v_{t,i}$ to each $j$
```

The analysis of Push-Sum builds on an understanding of the *diffusion speed* of Uniform Gossip, characterizing how fast a value originating with any node diffuses through the network.

Theorem 12.1. *The diffusion speed of Uniform Gossip is* $T_U(\delta, n, \epsilon) = 0(\log n + \log \frac{1}{\epsilon} + \log \frac{1}{\delta})$. *Thus, with probability at least* $1 - \delta$, *there is a time* $t = O(\log n + \log \frac{1}{\epsilon} + \log \frac{1}{\delta})$ *such that the contributions at all times* $t' \geq 1$ *and all nodes* i *are nearly uniform, i.e.,* $\max_j \left| \frac{v_{t',i,j}}{||v_{t',i}||_1} - \frac{1}{n} \right| \leq \epsilon$

It can be seen that choosing a neighboring giving 1/2 fraction to it makes the analysis easier. Push-sum is a very natural protocol, yet the proof of the approximation guarantee is non-trivial and relies crucially on a useful property, *mass conservation.* We also define the variance of the term $v_{t,i,j}$ as potential function.

12.2.4 Flooding

In several topologies, such as P2P or wireless radio networks, flooding seem to be more relevant. If the shares assigned to each neighbor are time independent, then we can characterize all shares by a matrix, $A = (\alpha_{i,j})_{i,j}$, where the entry $\alpha_{i,j}$ denotes what fraction of its vector $v_{t,i}$ node i sends to j. Let π denote the vector of stationary probabilities of the Markov Chain. Thus, $e_j^T.A^t$ is exactly the vector of contributions $(v_{t,i,j})$ from node j at all other nodes. For two vectors, a, b, we define the fraction a/b pointwise. To define the diffusion of flooding, we describe the following theorem:

Theorem 12.2. *Let T be a function such that* $\max_j ||\frac{e_j^T.A^t}{\pi} - 1||_2 \leq \epsilon$, *for all* $t \geq T(n, \epsilon)$. *Then* $T_F(n, \epsilon) := 2T(n, \sqrt{\frac{n\epsilon}{2+n\epsilon}})$ *is an upper bound on the diffusion speed for the flooding mechanism defined by A.*

For some graphs called *expanders*, $T_F(n, \epsilon) = O(\log n + \log \frac{1}{\epsilon})$.

12.2.5 Linear Synopses

Another applications of gossiping is in answering database queries that can be approximated well using linear synopses, i.e. functions h on multisets such that $h(S_1 \cup S_2) = h(S_1) + h(S_2)$. Other applications could be join size estimation, approximate histogram estimation and distinct value queries. The convergence is given by $T = kT_U(n, \epsilon)$.

Random sampling can be done as: For quantile queries, we can use mix of Push-Sum and

Algorithm 4 Random Sampling

```
1: Each node has a set of values of size $m_i$.
2: Initially, pick one uniformly at random.
3: Each step, send the sample and its weight ($m'$).
4: Time $T = T_U + O(\log n)$
```

Push-Random in loop. The convergence complexity is given by $T = O((\log m + \log(1/\delta)) \cdot (\log n + \log m + \log\log(1/\delta)))$.

12.3 Conclusion

Thus, we have seen a framework in which gossiping is presented as an way to compute different types of aggregate queries. The power of the approach also comes with liabilities. But the major concern is energy consumption. There are many open problems for future research. It is not clear how to stop gossiping, or how to compute continuous Top-K queries. Also, it would be desirable to develop techniques that allow nodes to estimate the current error of approximation without knowledge of the underlying network or communication mechanism.

Bibliography

[1] B. Baker and R. Shostak. Gossips and Telephones. pages 191–193, 1972.

[2] Alex Bavelas. *Communication Patterns In Task-Oriented Groups.* Dorwin Cartwright and Alvin Zander (eds.). Evanston, ll.: Row, Peterson & Company, 1950.

[3] F. Harary and A.J. Schwenk. The communication problem on graphs and digraphs. pages 491–495, 1974.

[4] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based Computation of Aggregate Information. In *Proceedings of Foundations of Computer Science*, Boston, 2003.

[5] R. Labahn. Some minimum gossip graphs. *Networks*, 23:333–341, 1993.