College of Computer & Information Science
Northeastern University
CS 7880: Network Algorithms

Spring 2016
2 February 2016
Scribe: Alp Elçi

- Gossip in Networks, Random Linear Network Encoding

1 Gossip in Networks

Let there be k distinct messages in a network. Gossip is the process of having these k messages being learned by all of the nodes. In each round, a node can broadcast one message to its neighbors. We will investigate how long it takes for the gossip to complete.

Note that in the model we consider the communication has no collisions. Also note that each node can simultaneously transmit and receive.

Example: Line Graph Assume that in a line graph G, all n nodes have exactly one message. The gossip in this case can be completed in $2n$ rounds, by sending the gossips to one direction for the first n rounds then sending the gossips to left in the next n rounds.

Example: Tree Gossip in a tree can be completed in at most $2n$. All nodes can first broadcast their messages up to the root. After the root has received all of the messages, it can progressively broadcast the gossips down.

Consider gossip for a network in which the edges arbitrarily change between after each timestep, and it is always connected. As a potential approach, the nodes can round-robin through their messages. For this approach, consider that all n nodes have learned $n-1$ messages, and there is one message, m, that is possessed only by the vertex, v. Since m would be broadcast on average once per n rounds, it would take $O(n^2)$ time for m to propagate through all n nodes.

In this lecture, we will see that there is a much more efficient way of completing gossip using the novel idea of network coding. Gossip for *arbitrarily changing networks* can be solved through random linear network coding (RLNC). The result we will present here is due to Haeupler [1].

Assume that there are $n = |V|$ messages in a network, G. Each message has a length of L, in bits. Let $\vec{m_i}$ be the i^{th} message.

$$m_i \in \mathcal{F}_2^L$$

Where $\mathcal{F}_2^L$ is the L-dimensional integral field in mod 2. For this section, all arithmetic operations will be conducted in base 2. In RLNC, the nodes will send packets instead of the raw messages. A packet is a tuple of $(\vec{\mu}, \vec{m})$, where $\vec{\mu}$ is the coefficient vector of $\vec{m}$, denoting linear combination coefficients of the n raw messages that span form $\vec{m}$.

$$\vec{m} \in \mathcal{F}_2^L$$

$$\vec{\mu} \in \mathcal{F}_2^n$$

For a low overhead, it is assumed that $L \approx n$.

Example: 4 nodes Assume the node v has packets with coefficient vectors: $(1,0,0,0)$, $(0,1,0,0)$. v can send packets with coefficient vectors of: $(1,0,0,0)$, $(0,1,0,0)$, $(1,1,0,0)$. The message of $(1,1,0,0)$ will be the $\oplus$ of the known two messages.

In RLNC, once a node has v linearly independent coefficient vectors, the node can reconstruct all of the messages. The sending protocol is as the following

- For each v, let Y_v denote the span of the coefficient vectors of packets stored by v. Pick a random vector in this span.
- Send the packet corresponding to the selected coefficient vector.

As a note, for choosing a random vector from Y_v, a random linear combination of the coefficient vectors received so far can be taken. The proof of this is left to the reader as practice.

Theorem 1. *$Pr[$gossip completes in $O(n)$ rounds$] \geq 1 - \frac{1}{2^{\Omega(n)}}$.*

Lemma 1. *If the span of the coefficient vectors received by v is $\mathcal{F}_2^n$, then v can decode all of the n messages by applying Gaussian elimination over the received messages.*

Initially, the Y_v. Since $\mathcal{F}_2^n$ is a finite space, Y_v can have at most 2^n vectors.

Definition 1. *v **knows about** $\vec{\mu}$ if $\vec{\mu}$ is not orthogonal to Y_v, hence $\exists \vec{x} \in Y_v, < \vec{x}, \vec{\mu} > \neq 0$.*

Lemma 1 can be reconstructed as the following.

Lemma 2. *If v knows about every vector in $\mathcal{F}_2^n$, then v can decode all n messages for $v \in V$, $\vec{\mu} \in \mathcal{F}_2^n$.*

As an important note, v knowing about all n singular vectors does not imply that v can reconstruct all of the n messages. v must know about all vectors in $\mathcal{F}_2^n$. As an example, if $n = 2$, a vector that has only one packet with $\vec{\mu} = (1,1)$ would know about both singular vectors and would be unable to reconstruct the messages.

Lemma 3. *If A knows about $\vec{\mu}$ and transmits something to v_B, with probability $1/2$, B will know about $\vec{\mu}$.*

This claim is particularly strong since it is applied simultaneously for the exponential number of $\vec{\mu}$s. Before giving the proof of Lemma 3, we show how Theorem 1 follows from Lemma 3.

Proof of Theorem 1 Fix any $\vec{\mu}$. Let S be the set of vertices that know about $\vec{\mu}$. Let U be the set of vertices that don't know about $\vec{\mu}$. By **Lemma 3**,

$$Pr[|S| \text{ will increase by } 1] \geq 1/2$$

Consider that $12n$ timesteps are elapsed. The probability that $|S|$ is not n is at most the probability that in $12n$ tosses of a fair coin, the number of heads is less than n. This probability can be upper bounded by a Chernoff bound as follows. The expected number of heads in $12n$ tosses of a fair coin is $6n$. So the probability that we have fewer than n heads is at most

$$e^{-(5/6)^2 \cdot 6n/3} \leq e^{-25n/18}.$$

$$Pr[\text{all nodes know of all } \vec{\mu} \text{ after } 12n \text{ rounds}] \geq 1 - \frac{2^n}{e^{25n/18}} \geq 1 - \frac{2^n}{4^n} = 1 - \frac{1}{2^n}.$$

□

Following is the proof of **Lemma 3**.

Proof of Lemma 3 Let A be the node that knows about $\vec{\mu}$. Let Y_A be partitioned as Y_A^0, such that $\forall \vec{c} \in Y_A^0$, $< \vec{c}, \vec{\mu} >= 0$, and Y_A^1, such that $\forall \vec{c} \in Y_A^1$, $< \vec{c}, \vec{\mu} >= 1$.

Let $\vec{\beta} \in Y_A^1$. Fix $\vec{\beta}$. $\forall \vec{\alpha} \in Y_A^1, \vec{\alpha} + \vec{\beta} \in Y_A^0$ since the sum will not be perpendicular to $\vec{\mu}$. Since Y_A^1 is not empty by the initial assumption, $|Y_A^1| > |Y_A^0|$.

□

References

[1] B. Haeupler. Analyzing network coding gossip made easy. In *ACM STOC*, 2011.