SIMPLE, FAST AND DETERMINISTIC GOSSIP AND RUMOR SPREADING

Bernhard Haeupler

(slides by Moshe Gabel)
Gossiping For Fun and Profit

• Want to spread a nasty rumor, but how?
  • Shouting from the top of a mountain?
  • Broadcast forbidden!

• In a distributed system, which nodes are up?
  • Just because I can’t reach some node, does not mean it is down.

• New node joined, must tell everyone.
The GOSSIP Model

• Unique IDs

• Neighbors and their UID are known

• In each **round**, each node can **exchange** messages with **one** neighbor
  • vs. all neighbors in LOCAL model

• Unlimited message size and local computation
Gossip Broadcast Algorithm

• Surprisingly easy.

• Repeat:
  • Choose neighbor uniformly at random
  • Exchange information

• $O\left(\frac{\log n}{\phi}\right)$ with high probability.
  • Fast on “nice” graphs.
  • And this bound is tight.
Gossip Example
Gossip Example
Gossip – Round 1

Graph representation of gossip propagation in Round 1.
Gossip – Round 1
Gossip – Round 2

1, 3

2, 3

4, 5, 7

4, 7, 8

5, 6, 7, 9

7, 8

1, 2, 3, 6

3, 6, 9

5, 6, 7, 9

4, 5, 9

1, 3

2, 3

4, 5, 7

4, 7, 8

7, 8

1, 2, 3, 6

3, 6, 9

5, 6, 7, 9

4, 5, 9
Gossip – Round 2

1, 2, 3, 4, 5, 9

1, 2, 3, 4, 5, 6, 9

4, 5, 7, 8, 9

4, 5, 6, 7, 8, 9

1, 2, 3, 6

1, 2, 3, 6

5

2, 3, 4, 5, 7, 8, 9

1, 2, 3, 6

3, 5, 6, 7, 9

3, 5, 6, 7, 8, 9

4, 5, 6, 7, 8, 9
Gossip – Round 3

1,2,3,6
4,5,7,8
4,5,7,8,9
1,2,3,4,5,9

5
2,3,4,5,7,8,9
1,2,3,6
3,5,6,7,9
3,5,6,7,8,9
4,5,6,7,8,9
Gossip – Round 3

Nodes 1, 2, 3, 4, 5, 6, 7, 8, 9

- Node 1: 1, 2, 3, 4, 5, 6, 9
- Node 2: 1, 2, 3, 4, 5, 6, 7, 8, 9
- Node 3: 1, 2, 3, 4, 5, 6, 7
- Node 4: 2, 3, 4, 5, 7, 8
- Node 5: 3, 4, 5, 6, 7, 8, 9
- Node 6: 1, 2, 3, 4, 5, 6, 7, 9
- Node 7: 2, 3, 4, 5, 6, 7, 8, 9
- Node 8: 3, 4, 5, 6, 7, 8, 9
- Node 9: 1, 2, 3, 4, 5, 6, 7, 8, 9
Gossip – Round 4

1, 2, 3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 9

2, 3, 4, 5, 7, 8

3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 7

1, 2, 3, 5, 6, 7, 9

2, 3, 4, 5, 6, 7, 8, 9

3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 7, 8, 9
Gossip – Round 4

1,2,3,4,5,6,7,9

1,2,3,4,5,6,7,8,9

1,2,3,4,5,6,7,8,9

1,2,3,4,5,6,7,8,9

1,2,3,4,5,6,7,8,9

1,2,3,4,5,6,7,8,9

1,2,3,4,5,6,7,8,9

1,2,3,4,5,6,7,8,9

2,3,4,5,6,7,8,9

1,2,3,4,5,6,7,8,9

1,2,3,4,5,6,7,8,9

1,2,3,4,5,6,7,8,9

1,2,3,4,5,6,7,8,9

1,2,3,4,5,6,7,8,9
Gossip – Round 4

1, 2, 3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 7, 9

1, 2, 3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 7, 8, 9

2, 3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 7, 8, 9
Gossip – Round 5

1, 2, 3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 7, 9

1, 2, 3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 7, 8, 9

1, 2, 3, 4, 5, 6, 7, 8, 9
Gossip – Round 5
Gossip – Round 5
Local Broadcast

• Some graphs have bottlenecks – small $\phi$.
  • And conductance is complicated and unintuitive.

• Simplify problem to 1-local broadcast: exchange rumor only with local neighborhood.

• Solvable w.h.p with non-uniform gossip $O(\log^3 n)$.

• Repeat $D$ (diameter) times $\Rightarrow$ get $O(D \log^3 n)$ for any graph.
Everything Is Randomized

• These algorithms succeed with high probability.

• Question: is randomness necessary? Can we guarantee success?

• Research has reduced needed random bits...

• But consensus was that some randomness is critical for efficient rumor spreading.
A Deterministic Approach

- Turns out a deterministic approach can be fast.

- A FLOOD primitive: spreads knowledge symmetrically in subset of local neighborhood.

- Exponentially growing structures under nodes.

- If local broadcast not over: structures disjoint.
Exponentially Growing Disjoint Structures

- Structures “under” nodes $u$ and $v$
Exponentially Growing Disjoint Structures

• Structures “under” nodes $u$ and $v$
Exponentially Growing Disjoint Structures

- Structures “under” nodes $u$ and $v$
Basic Notation

• \( \log x \): denotes \( \lfloor \log_2 x \rfloor \)

• \( n \): number of nodes in the graph

• \( R_v \): the set of nodes \( v \) has heard from
  • \( v \) knows the rumors of all \( u \in R_v \)

• Nodes exchange all rumors they know.
Round Robin Flooding

• Given a subgraph with known max degree $M$.

• Easy to spread rumor to $d$-neighborhood.

• Each node repeats $d$ times:
  • Spread rumors to all neighbors (max is $M$)

• Takes $dM$ rounds.
Round Robin Flooding

- Input: $m_v$ neighbors $u_1 \ldots u_{m_v}$ for each node $v$.
- Input: distance $d$, max degree $M = \max(m_v)$.

$R_v \leftarrow \{v\}$ (v initially knows only own rumor)

- Iterate d times:
  - For $t$ in $1 \ldots m_v$: exchange $R_v$ rumors with $u_i$
  - If $m_v < M$, wait $M - m_v$ rounds
  - Add new rumors from $u_1 \ldots u_{m_v}$ to $R_v$
Round Robin Flooding

Full graph
Round Robin Flooding

Subgraph
Round Robin Flooding

\[ M = 4, \ d = 3 \]
Round Robin Flooding

Let's look at who knows node $v$
Round Robin Flooding

Iteration 1, t = 1
Round Robin Flooding

Iteration 1, t = 2
Round Robin Flooding

Iteration 1, $t = 3$
Round Robin Flooding

Iteration 1, $t = 4$ (wait!)
Round Robin Flooding

Iteration 2 starts, let’s look at neighbors
Round Robin Flooding

Iteration 2 \( t=1 \)
Round Robin Flooding

Iteration 2 ends
Round Robin Flooding

Iteration 3 starts
Round Robin Flooding

Iteration 3 ends
Round Robin Flooding

Run ends
Round Robin Flooding Results

• $d$-neighborhood of $v$ knows rumor of $v$

• Equivalently: $v$ knows all rumors of its $d$-neighborhood

• Symmetry: $u \in R_v \iff v \in R_u$
Deterministic Gossip 1-Local Broadcast

- $R_v \leftarrow \{v\}$
- While not all $v$’s neighbors are in $R_v$:
  - Arbitrarily pick neighbor $u_i$ not already in $R_v$
  - Add $u_i$ (and edge) to subgraph $\{u_1 \ldots u_{i-1}\}$
  - FLOOD with $d = 2 \log n$ hops, $M = \log n$
  - Add rumors to $R_v$

- Nodes can terminate early (but still flood)
Correctness

• Trivial: algorithm stops only when \( v \) knows about all its neighbors.

• Hence all \( v 's \) neighbors know about \( v \).

• Symmetry will be important in efficiency proof.
  • It comes from the round robin flooding.
1-local broadcast

\[ n = 16 \rightarrow d = 2 \log n = 8, \quad M = \log n = 4 \]
1-local broadcast

End of iteration 0
1-local broadcast

After 2 iterations everyone knows about v
Deterministic Gossip Runtime

- Claim 1: at most $\log n$ iterations.

- Claim 2: total runtime at most $2 \log^3 n$ rounds.

- If true, already equivalent to random gossip!

- Implies $O(D \log^3 n)$ global broadcast.
Proof Sketch

• Define $i$-tree – a tree structure with $2^i$ nodes.

• Node $v$ has not terminated at iteration $i$?
  $\rightarrow$ there are $i$-trees rooted at $v$ and its uncontacted neighbors and tree of $v$ disjoint from others.

• $i$-Trees grow exponentially so there could only be max $\log n$ iterations.
\( i \)-tree: Binomial Tree with \( 2^i \) nodes

- **0-tree** is a single node which is the root.
- **\( i + 1 \)-tree** is two \( i \)-trees connected by edge, root is one of the two original roots.
Lemma

• At beginning of iteration \( i \), \( 0 < i < \log n \).

• \( H_i \) – graph of nodes/edges used thus far.
  • Subgraph used in flooding at iteration \( i \).

• If \( v_0 \) missing rumors from neighbors \( v_1 \ldots v_k \).

• Then there are \( k + 1 \) \( i \)-trees, \( T_0 \ldots T_k \subseteq H_i \), with roots \( v_0 \ldots v_k \), and \( T_0 \) disjoint from others.
Example

Start of iteration 0
Example

Start of iteration 1
Lemma Proof By Induction

- **Base:** $i = 0 \implies$ each node is 0-tree, all disjoint.

![Diagram of nodes $v_0$, $v_1$, $v_2$, $v_3$, $v_4$, $v_5$ with unused edges connecting them.]

Simple, Fast and Deterministic Gossip and Rumor Spreading

Slides by Moshe Gabel
Lemma Proof By Induction

• **Step:** assume $v_0$ is active in iteration $i + 1$

• $u_0$ is the neighbor chosen by $v_0$ in iteration $i$
Lemma Proof By Induction

• From induction hypothesis: at start of iteration \( i \), \( v_0 \) and \( u_0 \) were roots of two disjoint \( i \)-trees.

• Build \( T_0 \), an \( i+1 \)-tree rooted at \( v_0 \): connect \( i \)-tree of \( v_0 \) with \( i \)-tree of \( u_0 \) using edge \((v_0, u_0)\)
Lemma Proof By Induction

• Need to build $i + 1$-trees for $v_1 \ldots v_k$.

• From symmetry of flooding:

$$v_1 \ldots v_k \notin Rv_0 \iff v_0 \notin Rv_1 \ldots Rv_k$$

• Hence $v_1 \ldots v_k$ are active at iteration $i + 1$.

• Use same construction as $T_0$ to build $T_1 \ldots T_k$. 
Lemma Proof By Induction

• $T_0$ is disjoint from $T_1 \ldots T_k$ by contradiction.

• Assume $T_0$ and $T_j$ share node $w$ and $v_j \notin R_{v_0}$
Lemma Proof By Induction

• There is path $p = v_0 \ldots w \ldots v_j$.

• Tree depth $< \log n$ hence $\text{len}(p) \leq 2 \log n$
Lemma Proof By Induction

- There is path of length $\leq 2 \log n$ from $v_0$ to $v_j$.

- But rumors are flooded for $d = 2 \log n$.

- Hence $v_j \in R_{v_0}$ (that is, $v_0$ knows $v_j$).

- Contradiction $\Rightarrow T_0$ disjoint from $T_j$
Proof: $\log n$ iterations

- Assume not done after iteration $i = \log n$.

- At least two active nodes, and from Lemma two disjoint trees of depth $\log n$ as subgraphs.

- Each tree with $2^{\log n} = n$ nodes.

- Impossible since total nodes in graph is $n$. 

Proof: Runtime $O(\log^3 n)$ Rounds

- Each iteration adds one link.

- At most $\log n$ iterations $\Rightarrow M = \log n$

- FLOOD with $d = 2 \log n$, $M = \log n$ $\Rightarrow 2 \log^2 n$

- Total:
  
  $2 \log^2 n \times \log n$ iterations $= 2 \log^3 n$ rounds.
Wait a Second!

- Topology unknown but we need $\log(n)$.
- Start with estimate of $n$.
- Test completion with neighbor UIDs (known)
- If incomplete $\rightarrow$ square estimate and restart.
- Runtime $\text{polylog}(n) \rightarrow$ constant factor increase
- Sum of geometric series only constant factor of final successful run $\rightarrow$ still $\text{polylog}(n)$
Avoid Guesswork and Repetition

• Estimate-and-square strategy good for any algorithm with polylog time and simple completion check.

• Simpler strategy for deterministic gossip: flood with $d = 2i$ hops and $M = i$
  • At iteration $i$ each node used $i$ edges so $M = i$
  • $d$ is length of longest root-to-root path of two connected $i$-trees.
Summary

- Broadcast in GOSSIP model.

- **Deterministic**: always succeeds

- **Fast**: $O(\log^3 n)$ for 1-local or $O(D \log^3 n)$

- Can avoid FLOOD and get $O(D \log^2 n)$
  - And even $O(D + \log^{o(1)} n)$ with spanners
Flooding is Slow

• Why do we need flood?

• To make sure that neighbor picked in iteration \( i \) is from disjoint \( i \)-tree.

• To spread rumors symmetrically: all nodes in tree know root rumor and vice versa.

• Can do both with \( i \)-trees structure directly!
How An $i$-tree is Built

Sub-nodes keep adding nodes, but the $i$-tree for $\nu$ only contains nodes contacted up to time $i$. 
PUSH (Root Rumor Down The Tree)
PUSH (Root Rumor Down The Tree)
PUSH (Root Rumor Down The Tree)
PUSH (Root Rumor Down The Tree)
PUSH (Root Rumor Down The Tree)

At end, all children know root.
PULL (Node Rumors up to Root)

• PUSH was in decreasing order of edges: for $j = i$ down to 1: exchange rumors with $u_j$

• PULL is exactly like push in reverse order: for $j = 1$ to $i$: exchange rumors with $u_j$

• After PULL the root knows all rumors known to all children.
Deterministic Tree Gossip 1-Local B-cast

- $R_v \leftarrow \{v\}$
- While not all $v$’s neighbors are in $R_v$
  - Arbitrarily pick neighbor $u$ not already in $R_v$
  - $R' \leftarrow \{v\}, R'' \leftarrow \{v\}$
  - PUSH then PULL with $R'$
  - PULL then PUSH with $R''$
  - Add new rumors in $R'', R'$ to $R_v$
Lemma

• Claim: if $u$ doesn’t know neighbor $v$ at end of iteration $i$, their $i$-trees are disjoint.

Proof:

• If node $w$ belongs to two $i$-trees, it will know rumors of $u$ and $v$ after PUSH step.
• $u$ and $v$ will know all rumors of $w$ after PULL.
• After PUSH-PULL roots know about each other.
Deterministic Tree Gossip Runtime

• From same proof as before, \(\log n\) iterations:

• PUSH-PULL then PULL-PUSH gives symmetry.

• Lemma gives disjoint-ness of \(i\)-trees.

• Need \(4i\) rounds per iteration hence total

\[
\sum_{i=1}^{\log n} 4i = 2 \log n (1 + \log n) = O(\log^2 n)
\]
From 1-local to $k$-local Broadcast

- $k$-local becomes global if $k \geq D$

- Simulate any LOCAL algorithm of runtime $T$:
  - $T$ times $\times$ 1-local broadcast taking $T' = O(T' \cdot T)$
  - $\frac{T}{k}$ times $\times$ $k$-local broadcast taking $T'' = O \left( T' \cdot \frac{T}{k} \right)$

- Some LOCAL algorithms only need polylog neighborhoods
From 1-local to $k$-local Broadcast

- Initial iteration is just tree-building, actual 1-local broadcast achieved in the final iteration.

- First 1-local broadcast uses full algorithm.
- Remaining $k-1$ broadcasts repeat final iteration.

- Total: $2(\log n + \log^2 n) + (k - 1)2 \log n = 2(k \log n + \log^2 n)$ rounds.