Gossiping in a Multi-Channel Radio Network

Shlomi Dolev      Seth Gilbert      Rachid Guerraouï      Calvin Newport
Malicious Adversary + Radio = Dangerous
Malicious Adversary + Radio = Dangerous
How to circumvent the adversary?

Assume adversary cannot jam the channel.

- No collisions, no spoofing (e.g., [Koo 04]).

Assume adversary is probabilistic.

- Byzantine transmission fault occurs with probability $p < 1$ (e.g., [Pelc, Peleg 05]).

Assume adversary has limited broadcast power.

- Adversary can only broadcast $\beta$ times (e.g., [Gilbert, Guerraoui, Newport 06]).
Goal for Today

Remove restrictions on adversary:

1. Adversary can *jam* the channel.
3. *No bound* on adversarial broadcasts.
Byzantine Gossip in a Multi-Channel Radio Network

Multi-Channel Radio Network

Basic Model:

• $n$ nodes
• $c$ channels, synchronous
• Each can transmit or receive on 1 channel per round.
• Only 1 can transmit per channel per round.
• 1 adversary
• Adversary can disrupt 1 channel per round.
• If adversary disrupts channel, nothing is received.
Multi-Channel Radio Network

Example: $c = 2, \ n = 7$
Byzantine Gossip in a Multi-Channel Radio Network

Multi-Channel Radio Network

Example: \( c = 2, \ n = 7 \)
Example: $c = 2, \ n = 7$
Multi-Channel Radio Network

Example: $c = 2, \ n = 7$
Example: $c = 2, \; n = 7$
Gossip and $\epsilon$-Gossip

Basic Problem:

- Each node begins round with value $v_i$.
- Eventually, every node learns all $n$ values.

Gossip with $\epsilon$ error:

- All but $\epsilon n$ of the values are successfully transmitted to $n - 1$ nodes.
Randomized Algorithm for gossip:

Channel 1

Channel 2
Randomized Algorithm for gossip:

for $i = 1$ to $n$:

$\Pr(1/2)$

$v_i$

$\Pr(1/2)$

Channel 1

Channel 2
Randomized Algorithm for gossip:

for $i = 1$ to $n$:

$Pr(1/2)$

Channel 1

$Pr(1/2)$

Channel 2
Randomized Algorithm for gossip:

for $i = 1$ to $n$:

$\Pr(1/2)$

Channel 1

$\Pr(1/2)$

$\Pr(1/2)$

Channel 2

$\Pr(1/2)$

$\Pr(1/2)$
Randomized Algorithm for gossip:

for $i = 1$ to $n$:

- $Pr(1/2)$

$\nu_i$ → Channel 1

$Pr(1/2)$

$\nu_i$ → Channel 2
Randomized Algorithm for gossip:

for $i = 1$ to $n$:

$\mathcal{P}r(1/2)$

Channel 1

$\mathcal{P}r(1/2)$

Channel 2

$v_i$
Randomized Algorithm for gossip:

for $i = 1$ to $n$: 

- $\mathbb{P}r(1/2)$
  - $v_i$ to Channel 1
  - $\mathbb{P}r(1/2)$
  - Channel 2
Randomized Algorithm for gossip:

\[
\text{for } i = 1 \text{ to } n:
\]

\[
\mathbb{P}(1/2) \rightarrow \text{Channel 1} \rightarrow v_i
\]

\[
\mathbb{P}(1/2) \rightarrow \text{Channel 2} \rightarrow \cdots
\]
Randomized Algorithm for gossip:

for $i = 1$ to $n$:

$\Pr(1/2) \quad \text{Channel 1}$

$\Pr(1/2)$

$\Pr(\text{node } j \text{ receives value } v_i) = 1/4$
Randomized Algorithm for gossip:

for \( i = 1 \) to \( n \):

\[ Pr(1/2) \to \text{Channel 1} \]

\[ Pr(1/2) \to \text{Channel 2} \]

\[ Pr(\text{node } j \text{ receives value } v_i) = 1/4 \]

Repeat \( \Theta(\log n) \) times to achieve high probability.
Randomized Algorithm for gossip:

for $i = 1$ to $n$:

Channel 1

$Pr(1/2)$

$Pr(1/2)$

Channel 2

$Pr(node \ j \ receives \ value \ v_i) = 1/4$

Repeat $\Theta(\log n)$ times to achieve high probability.
Deterministic, Oblivious Algorithms
Deterministic, Oblivious Algorithms

Transmission pattern is determined prior to the beginning of the execution:

- Each node decides initially for each round $r$:
  a. whether to broadcast or receive in round $r$.
  b. which channel to use in round $r$. 
Deterministic, Oblivious Algorithms
Deterministic, Oblivious Algorithms

Observation: 0-Gossip is impossible.
Deterministic, Oblivious Algorithms

Observation: 0-Gossip is *impossible*.

- Adversary blocks node 1 every time it broadcasts:
Observation: 0-Gossip is impossible.

- Adversary blocks node 1 every time it broadcasts:
Deterministic, Oblivious Algorithms

Observation: 0-Gossip is impossible.

• Adversary blocks node 1 every time it broadcasts:
Deterministic, Oblivious Algorithms

Observation: 0-Gossip is impossible.

• Adversary blocks node 1 every time it broadcasts:
Deterministic, Oblivious Algorithms

Observation: 0-Gossip is *impossible*.

- Adversary blocks node 1 every time it broadcasts:
Deterministic, Oblivious Algorithms

Observation: 0-Gossip is impossible.

- Adversary blocks node 1 every time it broadcasts:

Node 1 never succeeds in broadcasting!
Gossip and $\epsilon$-Gossip

Basic Problem:

- Each node begins round with value $v_i$.
- Eventually, every node learns all $n$ values.

Gossip with $\epsilon$ error:

- All but $\epsilon n$ of the values are successfully transmitted to $n - 1$ nodes.
Outline

1. Introduction
2. Lower Bound
3. Algorithm for $\epsilon$-Gossip
4. Extensions
5. Open Questions
Lower Bound

Theorem 1:

Every deterministic, oblivious algorithm for $\epsilon$–Gossip requires at least

$$\Theta \left( \frac{n}{\epsilon c^2} \right)$$ rounds (for small $\epsilon$).

$n$ : number of nodes

$c$ : number of channels

$\epsilon$ : all but $\epsilon n$ values are successfully transmitted
Lower Bound

Theorem 1:

Every deterministic, oblivious algorithm for $\epsilon$-Gossip requires at least

$\Theta \left( \frac{n}{\epsilon c^2} \right)$ rounds.

Example:

- Gossiping all but one value requires $\Theta \left( \frac{n^2}{c^2} \right)$ rounds.
Lower Bound

Key Idea: (example: $c = 2$)

- If two nodes $a$ and $b$ never broadcast in the same round, the adversary can always block both of them.
  - Adversary disrupts node $a$ when $a$ broadcasts.
  - Adversary disrupts node $b$ when $b$ broadcasts.
- Thus disseminating all but one rumor requires at least $\binom{n}{2}$ rounds.
- Generalize for other values of $c$, $\epsilon$ . . .
Lower Bound

Consider the graph $G$:

- There is an edge between $i$ and $j$ if (and only if) nodes $i$ and $j$ never broadcast in the same round.

- If there exists a clique of size $k$ in $G$, then the adversary can prevent $k$ nodes from broadcasting.
Lower Bound

Consider the graph \( G \):

- There is an edge between \( i \) and \( j \) if (and only if) nodes \( i \) and \( j \) never broadcast in the same round.

Claim: If algorithm \( A \) solves \( \epsilon \)-Gossip in \( r \) rounds, then:

1. graph \( G \) has no clique larger than \( \epsilon n \).
2. graph \( G \) is missing at most \( c^2 r \) edges.
Lower Bound

Turan’s Theorem:

If graph $G$ has no clique of size $k+1$:
Then graph $G$ has at most:

$$\left(1 - \frac{1}{k}\right) \frac{n^2}{2}$$

edges.
Lower Bound

Corollary:

If graph $G$ has no clique of size $\epsilon n$:

Then graph $G$ is missing at least:

$$\binom{n}{2} - \left(1 - \frac{1}{\epsilon n - 1}\right) \frac{n^2}{2} = \frac{n^2}{2\epsilon n - 2} - \frac{n}{2} = \Theta \left(\frac{n}{\epsilon}\right) \text{ edges.}$$
Lower Bound

Theorem 1:

Every deterministic, oblivious $\epsilon$-Gossip algorithm requires at least $\Theta \left( \frac{n}{\epsilon c^2} \right)$ rounds.

Proof:

- Graph $G$ has no clique of size $\epsilon n$ and is missing at most $c^2 r \geq \Theta \left( \frac{n}{\epsilon} \right)$ edges, $r = \# \text{ of rounds}$. 

Turan’s Theorem
Outline

1. Introduction
2. Lower Bound
3. Algorithm for $\epsilon$-Gossip
4. Extensions
5. Open Questions
Upper Bound

Theorem 2:

There exists a deterministic, oblivious algorithm for $\epsilon$–Gossip that terminates in $\Theta \left( \frac{n}{\epsilon c^2} \right)$ rounds (for small $\epsilon$).

$n$ : number of nodes  
$c$ : number of channels  
$\epsilon$ : all but $\epsilon n$ values are successfully transmitted
Upper Bound

Construct the algorithm in two steps:

1. Data Collection
   - aggregate data at a small number of nodes.

2. Data Dissemination
   - broadcast data from a small number to the rest.
Case 1: $\epsilon \geq 1/c$

Data Collection:

Partition the nodes into two sets:

1. Listeners:
   - $2c$ nodes, two per channel.

2. Broadcasters:
   - Divided into $n/c$ groups of $c$ nodes.
   - Each group broadcasts in one round.
   - End result:
     
     All but $n/c$ are known to some pair of listeners.
Case 1: $\epsilon \geq 1/c$

Data Collection:

Partition the nodes into two sets:

1. Listeners:
   - $2c$ nodes, two per channel.

2. Broadcasters:
   - Divided into $\frac{(1-\epsilon)n}{c-1}$ groups of $c$ nodes.
   - Each group broadcasts in one round.
   - End result:
     
     At least $(1 - \epsilon)n$ are known to some listeners.
Case 2: $\epsilon < 1/c$

Data Collection

Partition the nodes into two sets:

1. Listeners:
   - $2c$ nodes, two per channel.

2. Broadcasters:
   - Divide into $\epsilon n$ sets of size $1/\epsilon$.
   - For each, all but 1 node succeeds in broadcasting.
   - End result: all but $\epsilon n$ values are known to a listener.
Case 2: \( \epsilon < 1/c \)

Data Collection

- For each set of size \( 1/\epsilon \):
  - Divide into \( 2/\epsilon c \) subsets of size \( c/2 \).
  - For every pair of subsets, assign 1 round.

- Result:
  - Only one node in the set can be blocked.
  - Running time: \( \binom{2/\epsilon c}{2} = \Theta \left( \frac{1}{\epsilon^2 c^2} \right) \).
Case 2: $\epsilon < 1/c$

Data Collection

- Overall running time (so far):

$$\epsilon n \cdot \Theta \left( \frac{1}{\epsilon^2 c^2} \right) = \Theta \left( \frac{n}{\epsilon c^2} \right)$$

- number of subsets
- rounds per subset
Data Collection Phase

Result:

- All but $\epsilon n$ are known to some pair of listeners.

Running time: $\Theta\left(\max\left(\frac{(1 - \epsilon)n}{c - 1}, \frac{n}{\epsilon c^2}\right)\right)$

Big $\epsilon$ Small $\epsilon$
Data Dissemination

Three steps:

• Step 1: Divide the nodes into $c$ sets, 1 per channel.
• Step 2: For each channel, disseminate data from 2 listeners to all the nodes in that channel’s set.
• Step 3: Merge channel sets pairwise.
Data Dissemination

Step 2:

- For each channel, disseminate data from 2 listeners to the nodes in that channel’s set.
Data Dissemination

Step 2:

- For each channel, disseminate data from 2 listeners to the nodes in that channel’s set.
Data Dissemination

Step 2:

- For each channel, disseminate data from 2 listeners to the nodes in that channel’s set.

\[ \log n \text{ rounds} \]
Data Dissemination

Step 2:

- For each channel, disseminate data from 2 listeners to the nodes in that channel’s set.

- Results:
  
  All but 1 node in the set receives all the values from the listeners.

  We say that all but 1 of the nodes are **knowledgable**; 1 node is **unknowledgable**.
Data Dissemination

Three steps:

- Step 1: Divide the nodes into $c$ sets, 1 per channel.
- Step 2: For each channel, disseminate data from 2 listeners to the nodes in that channel’s set.
- Step 3: Merge channel sets together.
Data Dissemination

Step 3:

- Merge channel sets together.
Data Dissemination

Step 3:

- Merge channel sets together.
Data Dissemination

Merge channel sets: $c \rightarrow c / 2$

- Choose 3 pairs (6 nodes) from each channel set.
  - At most 1 node in each channel set is unknowledgable.
  - For at least 1 pair of nodes, both are knowledgable.

- Pair the $c$ channel sets into $c / 2$ groups.
- For each group, for each of the 3+3=6 pairs of listeners, run disseminate routine from Step 2.
Upper Bound

Data Dissemination

- Step 1: Divide the nodes into $c$ sets, 1 per channel.
- Step 2: For each channel, disseminate data from 2 listeners to the nodes in that channel’s set.
  - Number of rounds: $\log n$

- Step 3: Merge channel sets together.
  - Number of rounds: $\log^2 n$
Upper Bound

Data Collection + Data Dissemination

- Running time (for small $\epsilon$):

$$\Theta \left( \frac{n}{\epsilon c^2} + \log^2 n \right) = \Theta \left( \frac{n}{\epsilon c^2} \right)$$
Upper Bound

Data Collection + Data Dissemination

- Running time (for small $\epsilon$):

$$\Theta \left( \frac{n}{\epsilon c^2} + \log^2 n \right) = \Theta \left( \frac{n}{\epsilon c^2} \right)$$

- Actual bound:

$$\max \left( \Theta \left( \frac{(1 - \epsilon)n}{c - 1} + \log_c n \right), \Theta \left( \frac{(1 - \epsilon)n}{\epsilon c^2} \right) \right)$$
Outline

1. Introduction
2. Lower Bound
3. Algorithm for $\epsilon$-Gossip
4. Extensions
   - Multi-Channel Adversary
   - Byzantine Corruptions
5. Open Questions
Multi-Channel Adversary

Assume the adversary can jam $t < c$ channels.

- More listeners.
- More rounds of Data Collection.
- Generalized Data Dissemination.
Multi-Channel Adversary

More Listeners:

- $t + 1$ listeners per channel
Multi-Channel Adversary

More Data Collection:

- Subdivision 1: $\epsilon n / t$ sets of size $t / \epsilon$.
- Subdivision 2: subsets of size $c / (t + 1)$.
- Schedule all \( \binom{t(t+1)}{\epsilon c} \) of the subsets to broadcast simultaneously.
Multi-Channel Adversary

More Data Collection:

• Running time:

\[ O \left( \frac{ne^{t+1}}{ce^t} \right) \]
Multi-Channel Adversary

Generalized Data Dissemination:

- Every set of $t + 1$ nodes is scheduled to listen together in one round.
  - Note: not just a base-$(t + 1)$ representation.

- Key tool: $(n, k, 1)$-selectors
  - Recursive construction: assume solved for $t$.
  - For each set in $(n, t, 1)$-selector:
    - Schedule that set on channel $t + 1$.
    - Use recursive construction to schedule $[1..t]$
Multi-Channel Adversary

Generalized Data Dissemination:

• Every set of $t + 1$ nodes is scheduled to listen together in one round.
  - Note: not just a base-$(t + 1)$ representation.

• Disseminate running time: $(t + 1)^{t+1} \log^t n$
Multi-Channel Adversary

Generalized Data Dissemination:

- Every set of $t + 1$ nodes is scheduled to listen together in one round.
  - Note: not just a base-$(t + 1)$ representation.

- Disseminate running time: $(t + 1)^{t+1} \log^t n$

- Total Data Disseminate running time:
  $$\frac{c \log c}{t + 1} (t + 1)^{t+1} \log^t n$$
Multi-Channel Adversary

Total running time:

$$O\left(\frac{ne^{t+1}}{c\epsilon^t} + c(t + 1)^t \log^{t+1} n\right)$$

** (Bound is tight when $\epsilon = t/n$.)
Byzantine Corruptions

Adversary can corrupt $t$ nodes.

- Not just jam channels.

- Main problem:
  - Some of the listeners may be corrupt.

- Solution:
  - More listeners: $(2t + 1)(t + 1)$
  - Run disseminate for each of the $(2t + 1)$ sets.
  - Only accept rumor if $(t + 1)$ listeners agree on it.
Open Questions and Ongoing Research

- Adaptive algorithms.
- Secure algorithms.
- Multi-hop networks.
- Energy efficiency and energy limited adversary.