Efficient adaptive dispensing against omission failures

Darek Kowalski, Michał Strojnowski
Plan

- Problem
- Model
- Failures
- Lower bounds
- Expanders
- Algorithm
- Conclusions
The gossip problem

$n$ processors
Each has its *rumor*

The goal is to exchange rumors, so that every processor get to know all the rumors

We would like to optimize time and number of messages sent
We focus here on deterministic solutions
Processors communicate by message passing

In one round each processor:
- receives messages
- makes local computation
- sends messages to chosen set of processors
Failures

- Up to $t$ processors may be faulty and omitting some incoming or outgoing messages.
- When some message do not reach its destination, either sender or receiver is faulty.

We now require only that correct processors exchange their rumors.
Solution

In two rounds:
1. Send your rumors to \( t+1 \) processors
2. Receive reply

\( 2(n-1)(t+1) \) messages

Can we do better?
Can we adapt to the number of failures that actually happens?
Lowering the number of messages

Define a *communication graph*, and send messages through its edges.

When most of the processors exchange information, deal with the rest.

We would like the communication graph to have a small number of edges and good connectivity.
Expander

- Degree of every node is at most $d$
- Each subset of $k < n/4$ nodes has $(1+c)k$ neighbors

Time is increased to $O(\log n)$, but communication cost drops
Isolated nodes

Some correct nodes become isolated by failures, and do not communicate.

To assure that node $p$ is faulty, we need $t+1$ processor that failed to communicate with $p$.

If $f$ nodes fails, we will send at least $O(tf)$ messages.
Lower bound

Thm. Each deterministic $t$-omission-fault-tolerant gossip algorithm may be forced to send $\max\{n, tf/4\}$ point-to-point messages in execution when $f$ failures happen.

- Proof by finding an execution when some correct node does not receive any message
Lower bound

Two cases indistinguishable from the point of view of every processor:
Communicators

- Degree of every node is $O(n^{\epsilon})$ (for constant $\epsilon$)
- For each $f < n/4$ and each subset $A$ of $n-f$ nodes, there exists subset of $A$ with $n-2f$ nodes that induces graph of diameter $2/\epsilon$

Proof of existence – by probabilistic argument
Mixing procedure

Exchange messages in communicator of degree $n^{\frac{1}{3}}$ for 6 rounds.

If $f < n/4$ then after this subroutine we have a set of $n-2f$ processors that exchanged rumors with each other.

We call such situation a rumor proliferation.
Testing failures

We choose set of \(9t\) processors. Each of them that knows at least \(n-9t\) rumors, asks for missing rumors.

(if \(n<9t\), all processors ask for missing rumors, and gossip is finished)

If rumors have previously been proliferated, it costs \(O(tf)\) messages
Basic algorithm

1. Mixing
2. Testing failures
3. Mixing
4. Each uninformed processor asks 9t for missing rumors and waits for answer

Constant time, $O(tf + n^{1+\frac{1}{3}})$ messages
Bipartite graph \((A,P,E)\), such that:

- \(|P|=n\), \(|A|>n^{1-\varepsilon}\)
- Degree of nodes in \(P\) is constant
- For \(f<n^{1-1.33\varepsilon}\), every subset of \(A\) with at least \(|A|-f\) nodes has at least \(|P|-f\) neighbors

Proof of existence – by probabilistic argument
Small-scale mixing

We take a set $A$ of $n^{3/4}$ processors, define a communicator $C$ with $\varepsilon=1/3$ for them, and connect them with the rest of processors using distributor $D$ for $\varepsilon=1/4$.

Procedure looks as follows:
1. Processors sends their rumors to neighbors in $D$
2. Set $A$ performs mixing using $C$
3. Set $A$ sends received info to neighbors in $D$

Constant time, $O(n)$ messages. If $f<n^{2/3}$, rumors are now proliferated.
Final algorithm

1. Small-scale mixing
2. If $t > n^{2/3}$, mixing for processors that collected less than $n - 4n^{2/3}$ rumors
3. Testing failures
4. Small-scale mixing
5. If $t > n^{2/3}$, mixing for uninformed processors
6. Each uninformed processor asks $9t$ for missing rumors and waits for answer

Constant time, $O(n + tf)$ messages.
Conclusions

Presented algorithm is up-to-constant optimal for this problem.

Future work:
- Explicit constructions of used graphs
- Using similar methods in crash-failure model
- Application to consensus problem