Distributed Fault-Tolerant Algorithms with Small Communication

Dariusz Kowalski
*University of Liverpool*

joint works with:

Bogdan Chlebus
*University of Colorado*

Chryssis Georgiou
*University of Cyprus*

Alex Shvartsman
*University of Connecticut & MIT*

Leszek Gasieniec
*University of Liverpool*

Piotr Musial
*University of Connecticut*
Overview

Model:
- synchronous, \( n \) processors
- message-passing full system, multicast

Processors failures (\( f \)):
- crashes,
- omissions,
- (authenticated) Byzantine

Problems:
- gossip,
- (adaptive) consensus
Fault-tolerant gossip

Initial configuration:
Each processor maintains an initial value, called a** rumor**

Terminal configuration:
Each processor has:
- All the rumors stored in non-faulty processors, and
- Either a **rumor** or a default value **faulty**, for every faulty processor
Gossip: known solutions

1. All-to-All:
   In the first round each processor sends its rumor to all other processors
   time: 1 round, communication: $\Theta(n^2)$
   resiliency: up to $n-1$ failures

2. With Leader:
   - Each processor sends its rumor to the leader
   - The leader combines received rumors and sends them back
   time: 2 rounds, communication: $\Theta(n)$
   resiliency: leader can not crash
Gossip: known solutions

3. With LeaderS:
   Repeat until all processors get a leader’s message
   - Each processor sends its rumor to the leader (next pid)
   - The leader combines received rumors and send them back
   
   time: $f+1$ rounds , communication: $\Theta(n(f+1))$
   resiliency: clean crashes only

4. Epidemic:
   Repeat until all rumors collected
   - A processor sends all known rumors to some processor
   
   time: $O(\log n)$ rounds , communication: $O(n \log n)$
   resiliency: tolerates linear number of failures
Consensus

Initial configuration:
Each processor stores an \textit{input value}, say binary from \{0,1\}

- \textbf{Termination:} Eventually every non-faulty processor decides on some decision value
- \textbf{Validity:} Any decision value is among the input values
- \textbf{Agreement:} The decision values of any two non-faulty processors are equal
Consensus: known solutions

1. Simple:
   - in the first round a processor broadcasts its initial value
   - after receiving a bigger value a processor broadcasts it
   - after $f+1$ rounds a processor decides on the maximum value

time: $f+1$ rounds (not early-stopping)
communication: $O(n^2)$
Consensus: known solutions

2. Diffusion tree:
   there is a conceptual balanced tree, agreement is made on every level of the tree and propagated up
   (i) time: $O(f+1)$
       communication: $O(n^{1+\varepsilon})$
   (ii) time: exponential
       communication: $O(n)$ communication bits!
Goal

Goal: complexity efficient algorithms

Gossip:
– time: asymptotically close to optimal $O(1)$
– message complexity: reducing to $O(n \text{ polylog } n)$ for $n$ processors

Consensus:
– time: asymptotically optimal $O(f)$ for $f$ crashes
– message complexity: reducing to $O(n \text{ polylog } n)$ for $n$ processors
How to achieve

Methods:
• Fault-tolerant gossip to spread information
  – Problem: How to exchange information in the presence of crashes:
    • quickly
    • with small number of messages
• Consensus: decision made by a majority that exchanges messages efficiently
  – Problem: How to define a majority which cooperates efficiently and eventually makes a decision
Distributed setting and complexity

- There are $n$ synchronous processors that communicate by sending messages (message-passing model)
- Communication is in rounds containing: local computation, sending messages, receiving messages
- A processor can send a message to any subset of processor in one round (multicast operation)
- During the execution $f$ processors crash ($f$ is unknown for the algorithm)

- Time: number of rounds by termination
- Message complexity: total number of messages sent

- Early-stopping consensus: in time $f + 1$
- Adaptive consensus: in time $O(f)$
Gossip: prior work (complexity)

Fault-tolerant Gossip:

- Time $O(\log^2 n)$, communication $O(n \log^2 n)$ in case $n - f = \Omega(n)$ [Chlebus,Kowalski SPAA’02]
- $\Omega(\log n / \log \log n)$ time required if communication $O(n \text{polylog } n)$
- Time $O(\log^2 n)$, communication $O(n^{1+\varepsilon})$, for any $0 < \varepsilon < 1$ [Georgiou,Kowalski,Shvartsman DISC’03]
- Time $O(\log^3 n)$, communication $O(n \log^4 n)$ [Chlebus,Kowalski DISC’06]
Consensus: prior work (complexity)

Consensus:

- Time complexity $f + 1 = \Theta(f)$ is optimal [Fischer,Lynch IPL’82]
- Adaptive solution with $O(n + fn^\varepsilon)$ communication complexity, for any $0 < \varepsilon < 1$ [Galil, Mayer and Yung FOCS’95]
- (Non-adaptive) solution $O(n)$ time and $O(n \log^2 n)$ communication complexity in case $n - f = \Omega(n)$ [Chlebus,Kowalski SPAA’02]
- Adaptive solution with $O(n \log^5 n)$ communication complexity [Chlebus,Kowalski DISC’06]
Gossip algorithm: preliminaries

Local structures maintained by processor $p$:
- Global *communication graphs* $G_i$ for $i = 1, \ldots, \log n$, of max degree $2^i \text{polylog } n$ (the same graphs for all processors)
- Local permutation $\pi_p$ of all other processors
- Local list $R_p$ of rumors (or faulty status) of other processors

Messages sent by processor $p$ contain:
- Local list $R_p$ of rumors (or faulty status) of other processors

Additional rule (*):
- If processor $p$ learns that $q$ has crashed than it sets status of $q$ in its local list $R_p$ to faulty, despite if it has rumor of $q$ or not
Gossip algorithm: gathering

PART I: Gathering

Iterate epochs $i = 1, \ldots, \log n$

Each epoch contains fixed polylog $n$ number of rounds;
in every round of epoch $i$ processor $p$:

- exchanges messages with its neighbors in communication graph $G_i$ (those that are not known by $p$ to be faulty: by checking $R_p$)
- sends requests to first $2i$ polylog $n$ processors from its local permutation, which status is unknown according to $R_p$
- answers to all processors that requested $p$ in the previous round

Crucial property after some epoch (follows from the properties of communication graphs and local permutations):

- There is an epoch after which there is a fraction of non-faulty processors such that each of them has filled its local list $R$
Gossip algorithm: spreading

PART II: Spreading

Exactly the same as PART I!

Well, not exactly - processor $p$ uses additional local list $S_p$ which plays the role of $R_p$, and status of $q$ on list $S_p$ means now
- informed (about all rumors/crashes), or
- faulty

Crucial property after some epoch (follows from the properties of communication graphs and local permutations):
- There is an epoch after which there is a fraction of non-faulty processors such that each of them has filled its local list $S$
Gossip algorithm: \textit{finally}

FINAL ALGORITHM

Every processor $p$
- Runs PART I to allow (fractional) gathering of rumors
- After filling its local list $R_p$ it initializes list $S_p$ and “switches” to PART II to spread its local (filled!) list of rumors to others;
  If two received lists $R_q$ and $R_r$ contain slightly different information:
  - one contains a rumor of some processor $z$ while the other has its status faulty
then status of $z$ in $R_p$ is faulty (by general rule (*) )
Example
Why and how it works?

Complexity:
Time:
\((\log n \text{ epochs}) \times (\text{polylog } n \text{ rounds}) = O(\text{polylog } n)\)

Messages:
\[\sum_{i \leq \log n} (2^i \text{ polylog } n) = O(n \text{ polylog } n)\]

Correctness:

• There is an epoch \(i\) such that:
  – There are at most \(n/2^i\) non-faulty processors at the beginning
  – There are at least \(n/2^{i+1}\) non-faulty processors at the end

• For this epoch both PARTs are working as expected (assuming efficient selection of communication graph and permutations), with the same fraction of non-faulty processors, so they successfully gather and spread the rumors in this epoch
Expander

Regular $n$-node graph $G=(V,E)$ is called an $a$-expander iff for every set $A \subseteq V$ of size greater than $a$, the set $N(A)$ of neighbors of $A$ in $V$ is of a size larger than $n - a$.

Existential: There exists $n$-node $a$-expander with degree $O((n/a) \log (n/a))$

Fact [Ta-Shma,Umans,Zuckerman, STOC’01]

An $n$-node $a$-expander with degree $O((n/a) \text{ polylog } n)$ is constructible in time polynomial in $n$, for any $n$ and $a < n$. 
Communication graphs

Define $G_i$ as $n/2^i$-expander.

Crucial fault-tolerant property of $G_i$:

- After removing all but $n/2^i$ nodes in graph $G_i$, there is a fraction of remaining nodes which constitutes a connected component of diameter $\Theta(\log n)$

[Chlebus, Kowalski, Shvartsman STOC’04]
Local permutations

**Constructed** the family of $n$ permutations on set \([n]\) having the following property:

- If a set of nodes cooperate to solve $n$ tasks, using unique permutations from the constructed family for selecting tasks, the number of task-performances is $O(n \text{polylog } n)$.

  [Kowalski, Musial, Shvartsman ICDCS’05]

**Existential** result: $\text{polylog } n = \log n$

[Anderson, Woll STOC’91] [Georgiou et al. DISC’03]

In the application to the gossip problem:

- task $j = \text{sending request msg to } j^{th}$ processor
Algorithm for *Consensus* (non-adaptive version)

Maintained local data:
- Initial value (0 or 1)
- Preference (0 or 1) initialized into initial value
- Status (hesitant or convinced), initially hesitant

Maintained global data:
- Communication graphs $G_i$ for $i = 0, 1, \ldots, \log n$, of max degree $2^i \text{polylog } n$
Algorithm for Consensus (non-adaptive version)

Algorithm
Iterate epochs \( i = 1, \ldots, \log n \)

- **Gossiping** the preferences by hesitant processors; convinced only reply for requests; preference set to the maximum preferences heard
- **Flooding** the preferences: for \( n/2^i \) rounds, if 1 received for the first time then broadcast it to all non-faulty neighbors
- **Checking**, for each processor if it is compact: for \( \Theta(\log n) \) rounds broadcast received pid’s to non-faulty neighbors, initially its own pid; compact if received a fraction of \( n/2^i \) of different pid’s

Hesitant and compact processor becomes convinced; hesitant and not compact resets its preference

Each processor decides at the end on its final preference
Algorithm for Consensus Analysis

Complexity:

Time:
\[ \sum_{i \leq \log n} \left( \frac{n}{2^i} + \text{polylog } n \right) = O(n) \]

Messages:
\( (\log n \text{ epochs}) \times (n \text{ polylog } n) = O(n \text{ polylog } n) \)

Correctness:

• Processor \( p \) decides 1  \( => \)
  \( p \) becomes convinced in some epoch \( i \) (\textit{checking}) \( => \)
  all convinced in epoch \( i \) decide on 1 (\textit{flooding}) \( => \)
  all others must decide on 1 (\textit{gossiping})
Remarks and open problems

• Consensus algorithm using optimal $O(n)$ communication bits, but works in exponential time
  [Galil, Mayer and Yung FOCS’95]

OPEN PROBLEMS:

• What about the other models of failures?
• What about tradeoff between time and communication complexities?