Fault-Tolerant Aggregation: Flow Update Meets Mass Distribution

Carlos Baquero
HASLab, INESC Tec, Universidade do Minho
cbm@di.uminho.pt

Joint work with (alpha order):
Paulo Sérgio Almeida, Martín Farach-Colton, Paulo Jesus, Miguel Mosteiro
Universidade do Minho, Rutgers University

OPODIS, December, 2011
Data Aggregation

Motivation - Distributed Data Aggregation

- Important building block of distributed applications
- Transmission of raw data may not be scalable/economic
- Raw data may not fit in memory footprints
- Raw data may be highly redundant
- Applications often rely on data summaries
To obtain network statistics and administration information:

- Network size
- Total resources available
- Average session time
- Max./Min. network load
- ...

Monitor and control a covered area (WSN):

- Min./Max. temperature
- Average humidity
- Concentration of a toxic substance
- Noise level
- ...

Several classes of aggregation algorithms

- Hierarchic: TAG
- Sketches: FM-Sketches, Extrema Propagation
- Sampling: Randomized Reports, Sample & Collide
- Averaging: Push-Sum, Push-Pull

Both Hierarchic and Averaging allow high precision aggregates.
Existing Approaches

Averaging
(e.g. Push-Sum/Synopses, Push-Pull, DRG)

- Iterative Averaging process
- Topology independent
- Result produced at all nodes

Correctness ⇒ “mass conservation”:

\[ \sum_{i \in V} v_i = \sum_{i \in V} e_i^t = k \]
Existing Approaches

Averaging
(e.g. Push-Sum/Synopses, Push-Pull, DRG)

- Iterative Averaging process
- Topology independent
- Result produced at all nodes

Correctness ⇒ “mass conservation”:

$$\sum_{i \in V} v_i = \sum_{i \in V} e_i^t = k$$

$\sum = 10$
$t = 1$
$0$
$3$
$3$
$3$
$1$
**Existing Approaches**

**Averaging**  
(e.g. Push-Sum/Synopses, Push-Pull, DRG)

- Iterative *Averaging* process
- Topology independent
- Result produced at all nodes

**Correctness**  \(\Rightarrow\) “mass conservation”:

\[
\sum_{i \in V} v_i = \sum_{i \in V} e_i^t = k
\]
Existing Approaches

**Averaging**
(e.g. Push-Sum/Synopses, Push-Pull, DRG)

- Iterative *Averaging* process
- Topology independent
- Result produced at all nodes

**Correctness** ⇒ “mass conservation”:

\[
\sum_{i \in V} v_i = \sum_{i \in V} e_i^t = k
\]
Message loss:

Problem:

- Loss of mass
- Violation of "mass conservation"
- Convergence to a wrong value

\[ \sum = 10 \quad t = 0 \]
Message loss:

Problem:

- Loss of mass
- Violation of “mass conservation”
- Convergence to a wrong value
Message loss:

Problem:
- Loss of mass
- Violation of "mass conservation"
- Convergence to a wrong value

\[ \sum_{t=2} = 9 \]
Message loss:

Problem:
- Loss of mass
- Violation of “mass conservation”
- Convergence to a wrong value

\[
\sum_{t=\infty} = 9
\]
Flow Updating (DAIS 2009, IEEE SRDS 2010)

Robust aggregation algorithm for dynamic networks:
- Independent from the routing topology (gossip-based)
- Converges to the correct result at all nodes
- Supports message loss and node crashes
- Self-adapts to changes in input values
Based on the concept of flow from graph theory
  - Examples: water flow, electrical current
Each node maintains flows to neighbors
  - Flows converge to symmetrical values
Update flows by sending idempotent messages
Keep the initial input values unchanged
Compute the aggregate from the initial value and flows
Flow Updating. Works very in empirical evaluations, but:

- Convergence in not monotonic
- Protocols have eluded analysis

MDFU was developed: Mass Distribution with Flow Updating

- Monotonic convergence
- Bounds for fault free and for stochastic message loss
Synchronous rounds

- Undirected connected graph \( G(V,E) \)
- For \( i \in V \): neighbors \( N_i \), degree \( |N_i| \)
- For \( (i,j) \in E \): \( D_{ij} = \max\{|N_i|, |N_j|\} \), \( \Delta = \max_{i \in V} |N_i| \)
- For edge and round, message failure probability \( f \)
Each node holds inputs value $v_i$

Each node needs to compute $\bar{v} = \sum_{i=1}^{n} v_i / n$

No global knowledge
- $n$ is unknown
- Nodes only know: $N_i$ and $D_{ij}$
- Nodes compute $e_i$, a local estimate of $\bar{v}$

Nodes track for each neighbor:
- inbound flow in $F_{in}$
- outbound flow in $F_{out}$

Nodes compute $e_i$, a local estimate of $\bar{v}$

$e_i = v_i + \sum_{j \in N_i} (F_{in}(j) - F_{out}(j))$
Flow Updating

MDFU Algorithm

// initialization

e_i ← v_i;

do foreach j ∈ N_i do
    F_{in}(j) ← 0;
    F_{out}(j) ← e_i / (2D_{ij});
end

// communication phase

do foreach j ∈ N_i do
    foreach j ∈ N_i do
        Send j message ⟨i, F_{out}(j)⟩;
    end
    F_{in}(j) ← F_{out}(j);
end

// computation phase

e_i ← v_i + \sum_{j \in N_i} (F_{in}(j) - F_{out}(j));

do foreach j ∈ N_i do
    F_{out}(j) ← F_{out}(j) + e_i / (2D_{ij});
end
end
Analytical Results

- For any target precision $0 < \xi < 1$
- Estimates within $[(1 - \xi)\bar{v}, (1 + \xi)\bar{v}]$
- For $f = 0$, a graph of size $n$ and conductance $\Phi(G)$
- Convergence time (in rounds) is at most: $2 \times \ln \frac{n}{\xi} \times \frac{1}{\Phi(G)^2}$
- Precision grows exponentially with rounds
- Under $f > 0$ we derived multiplicative overheads over $f = 0$
Comparative Results (no message loss)

Erdos-Renyi Random Network $G(V,E)$, $|V| = 1000$, $|E| = 5000$

- DRG and Push-Synopses will not tolerate message loss

- $CV(RMSE) = \sqrt{\sum_{i \in V} (e_i - \overline{v})^2 / n / \overline{v}}$
MDFU under message loss $f$

Erdos-Renyi Random Network $G(V, E), |V| = 1000, |E| = 5000$

- Convergence, not to $\bar{v}$, but to a bias point in $[(1 - f)\bar{v}, \bar{v}]$
- Mass is added as sent, in $F_{out}$, but not received in $F_{in}$
MDFU with Linear Prediction

- As node estimates converge, in neighbor nodes $e_x \approx e_y$
- Since flow $F_{out}$ increases by $e_i / (2D_{ij})$
- In each edge flow growth velocities also converge
- When messages are not received, in an edge, one can (linearly) predict how $F_{in}$ should have grown in an edge
- just multiply the last increase by the rounds with no message
- results are surprisingly good . . .
MDFU-LP under very high message loss $f$

Erdoes-Renyi Random Network $G(V, E)$, $|V| = 1000$, $|E| = 5000$

$$CV(RMSE) = \sqrt{\sum_{i \in V} (e_i - \bar{v})^2 / n / \bar{v}}$$
MDFU-LP with input value variation

- Most averaging algorithms (PUSH-*) distribute mass
- They restart when node input value $v_i$ changes
- Ongoing convergence is lost
- A recent exception is LiMoSense (ALGOSENSOR’11)
- MDFU (MDFU-LP) handle $v_i$ variations with no modifications
MDFU-LP with input value variation

Erdoes-Renyi Random Network $G(V, E)$, $|V| = 1000$, $|E| = 5000$, $f = 10\%$. 

![Graph showing the estimated values and real value over rounds](image-url)
High precision distributed aggregation requires averaging

Traditional “mass” exchange approaches should give way to idempotent algorithms, like *Flow Updating*

Future work can include.
- Asynchrony in MDFU (already studied in FU)
- More complex aggregates: Cumulative Distribution Functions
- Strategies to further increase convergence speed