Computing Aggregate Functions in Sensor Networks

Antonio Fernández Anta[1] Miguel A. Mosteiro[1,2] Christopher Thraves[3]

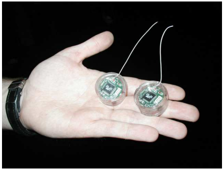
[1]LADyR, GSyC,Universidad Rey Juan Carlos

[2]Dept. of Computer Science, Rutgers University

[3]IRISA/INRIA Rennes

PRDC 2009

A Sensor Network

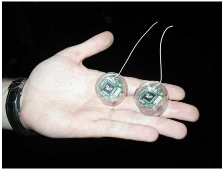


Intel Berkeley Research Lab

Capabilities
- processing
- sensing
- communication

Limitations
- range
- memory
- life cycle

A Sensor Network



Intel Berkeley Research Lab

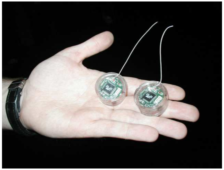
Capabilities

- processing
- sensing
- communication

Limitations

- range
- memory
- life cycle

A Sensor Network



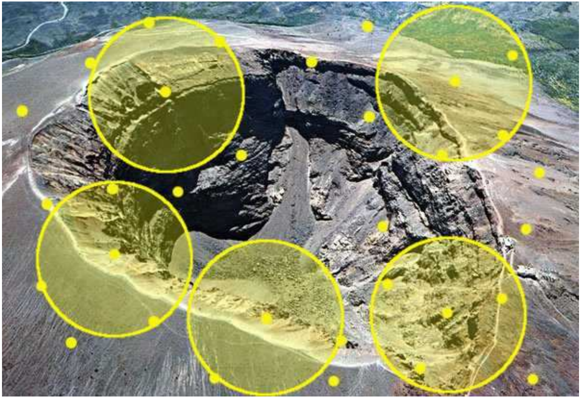
Intel Berkeley Research Lab

Capabilities

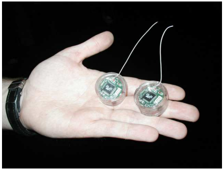
- processing
- sensing
- communication

Limitations

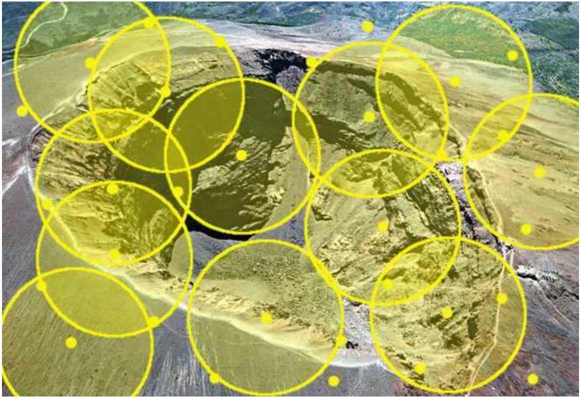
- range
- memory
- life cycle



A Sensor Network



Intel Berkeley Research Lab



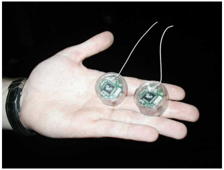
Capabilities

- processing
- sensing
- communication

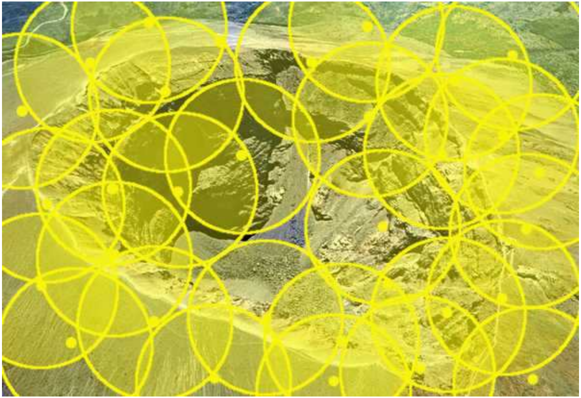
Limitations

- range
- memory
- life cycle

A Sensor Network



Intel Berkeley Research Lab



Capabilities

- processing
- sensing
- communication

Limitations

- range
- memory
- life cycle

The Problem

Node gets *input-value* (sensed, measured, etc.)

- unreliability ⇒ can not rely on individual sensors data ⇒ aggregate!

Algebraic aggregate functions:

- average
- maximum, count, sum, quantiles, etc. (easy from average [KDG03])

What average?

- lack of position information ⇒ aggregate all.
- sink nodes must receive ⇒ result to all nodes.
- input-values change over time ⇒ need global synch.
- multi-hop ⇒ impossible to aggregate in one step.
- under arbitrarily failures ⇒ aggregation is intractable! [BGMGM03]

Problem

Compute the average among all-nodes input-value at a given time step and distribute the result to all nodes under bounded failures.

Connectivity

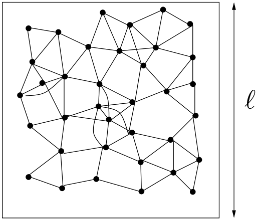
Node Deployment in Sensor Networks

- Hostile or remote environment

  $\Rightarrow$ deterministic deployment not feasible

  $\Rightarrow$ *controlled* random deployment.

- Arbitrary Density: the Geometric Graph $\mathcal{G}_{n,r}$.



- $[0,1]^2$
- Structural properties depend on relation among r and n.
- Connectivity/coverage guarantee.

Node Constraints

- Constant memory size.
- Limited life cycle.
- Short transmission range.
- Low-info channel contention:
  - Radio tx on a unique shared channel.
  - No collision detection.
  - Non-simultaneous rx and tx.
- Local synchronism.
- Discrete tx power range.
- No position information.
- Unreliability.
- Adversarial wake-up schedule.
- No global controller.
- No initial infrastructure.

tx = transmission.
rx = reception.

Other

- Failures: $\leq f$ failures separated $\geq T$ steps.
- Input values distribution: adversarial.
- Topology knowledge: unknown except for n.
- Failure-free *sink* node, knows D and Δ.
- ID: unique of $O(\log n)$ bits.
- Metrics:
  - time $\rightarrow$ slots.
  - energy $\rightarrow$ transmissions.

Previous Work

- Hierarchical Aggregation: tree convergecast.
  - Gupta et al. 01: $O(\log^2 n)$ *rounds*, no contention resolution.
  - Kollios et al. 05: $\omega(\log n)$ memory.
  - Madden et al. 02
- Non-hierarchical Aggregation: mass distribution.
  - Boyd et al. 06: prob $O(\log n + \log(n/\epsilon)/(1-\lambda_{max}((\mathbf{I}+\mathbf{P})1/2)))$ *rounds*.
  - Kempe et al. 03: similar bounds, one hop.
  - Chen et al. 06: prob $O(\Delta^3 \log(\sum_i (v_i - \overline{v})^2/\epsilon^2)/a(G))$ *rounds*.
  - ALL: $\omega(\log n)$ memory, no contention resolution, synch start.
- Geographic.
  - Dimakis et al. 08: needs position information.

Previous Work

- Hierarchical Aggregation:

  pros fast.

  cons failures → network partitioned.
  limited memory → can not be implemented.

- Non-hierarchical Aggregation:

  pros more resilient to failures.

  cons higher energy consumption.

- Our protocol:

  interleave both! with limited memory and low energy consumption.

Protocol

- **Preprocessing**
  - Partition nodes in *delegates* and *slugs*.
  - Reserve blocks of time steps for local use.
- **Aggregate Computation Scheme**
  - **Trigger:** sink broadcast (τ_1, Δ, D).
  - **Collection:** delegates aggregate slugs input value.
  - **Computation:** delegates compute aggregate function.
  - **Dissemination:** delegates distribute the result.

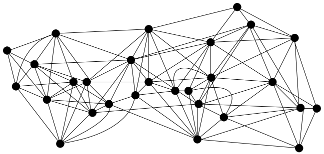
Protocol

1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
     - every slug is at $d \leq \alpha r$ from some delegate $(0 < \alpha \leq 1/4)$
     - every pair of delegates are at $d > \alpha r$

Protocol

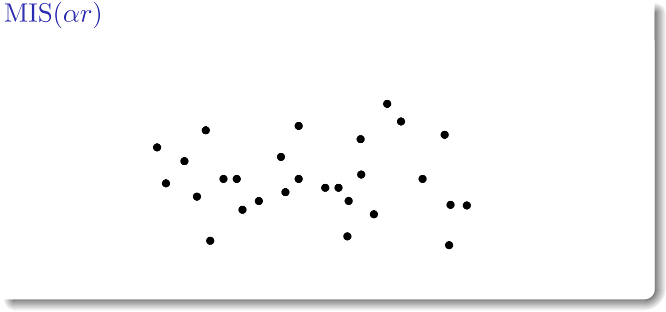
1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
     - every slug is at $d \leq \alpha r$ from some delegate ($0 < \alpha \leq 1/4$)
     - every pair of delegates are at $d > \alpha r$

MIS(αr)



Protocol

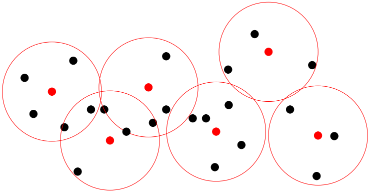
1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
     - every slug is at $d \leq \alpha r$ from some delegate $(0 < \alpha \leq 1/4)$
     - every pair of delegates are at $d > \alpha r$



Protocol

1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
     - every slug is at $d \leq \alpha r$ from some delegate $(0 < \alpha \leq 1/4)$
     - every pair of delegates are at $d > \alpha r$

MIS(αr)



Protocol

1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
     - every slug is at $d \leq \alpha r$ from some delegate $(0 < \alpha \leq 1/4)$
     - every pair of delegates are at $d > \alpha r$
   - Every delegate reserves blocks of time steps for local use.

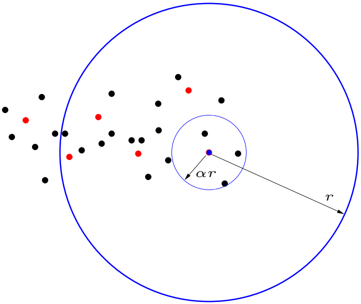
     s.t. delegate and slugs can communicate without collisions.

Protocol

1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
     - every slug is at $d \leq \alpha r$ from some delegate ($0 < \alpha \leq 1/4$)
     - every pair of delegates are at $d > \alpha r$
   - Every delegate reserves blocks of time steps for local use.

     s.t. delegate and slugs can communicate without collisions.

Coloring(r)

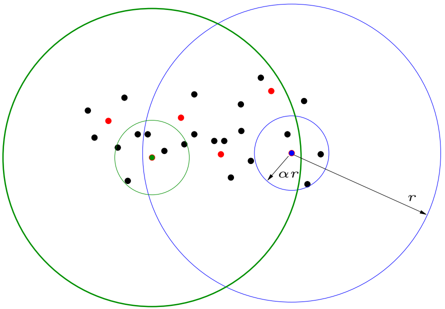


Protocol

1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
   - Every delegate reserves blocks of time steps for local use.

     s.t. delegate and slugs can communicate without collisions.

Coloring(r)



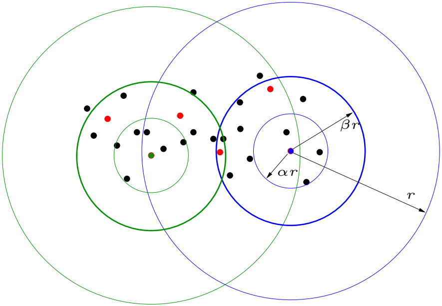
$$2\alpha \leq \beta \leq r/2$$

Protocol

1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
   - Every delegate reserves blocks of time steps for local use.

     s.t. delegate and slugs can communicate without collisions.

Coloring(r)

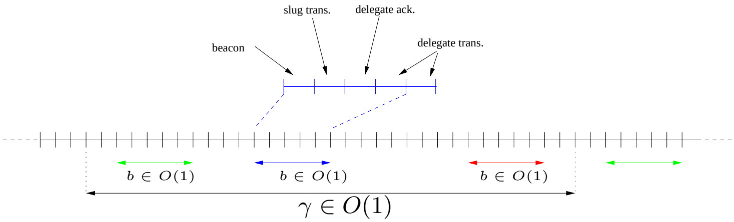


$$2\alpha \leq \beta \leq r/2$$

Protocol

1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
   - Every delegate reserves blocks of time steps for local use.

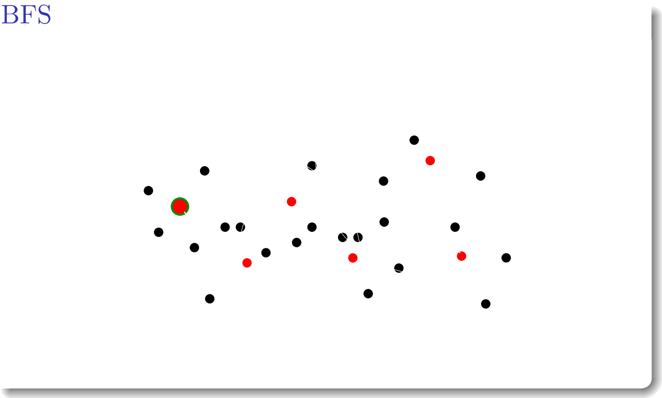
     s.t. delegate and slugs can communicate without collisions.



From now on, delegates use βr and slugs αr, in reserved slots.

Protocol

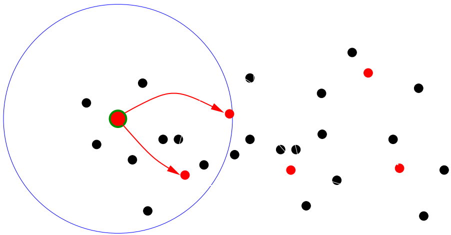
1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
   - Every delegate reserves blocks of time steps for local use.
2. **Trigger:** delegates flood τ_1 and define tree, starting from sink.



Protocol

1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
   - Every delegate reserves blocks of time steps for local use.
2. **Trigger:** delegates flood τ_1 and define tree, starting from sink.

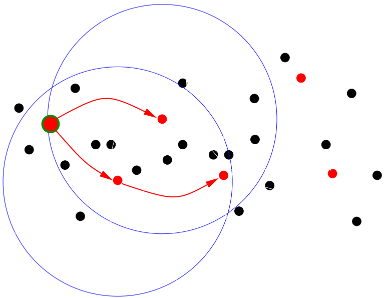
BFS



Protocol

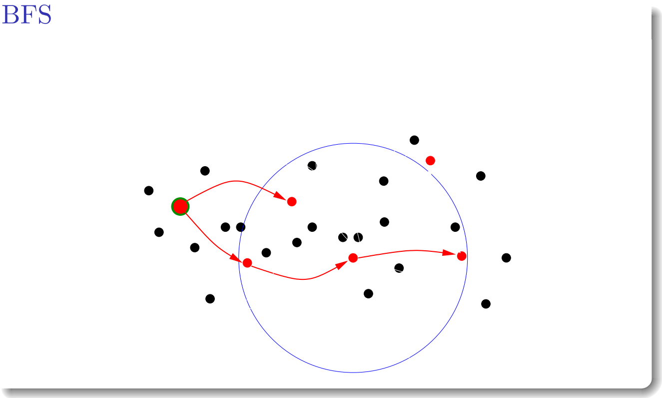
1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
   - Every delegate reserves blocks of time steps for local use.
2. **Trigger:** delegates flood τ_1 and define tree, starting from sink.

BFS



Protocol

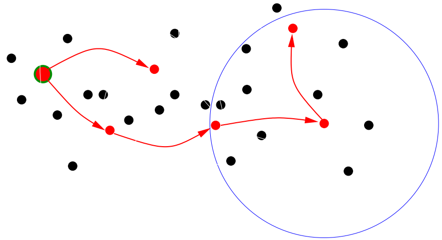
1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
   - Every delegate reserves blocks of time steps for local use.
2. **Trigger:** delegates flood τ_1 and define tree, starting from sink.



Protocol

1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
   - Every delegate reserves blocks of time steps for local use.
2. **Trigger:** delegates flood τ_1 and define tree, starting from sink.

BFS



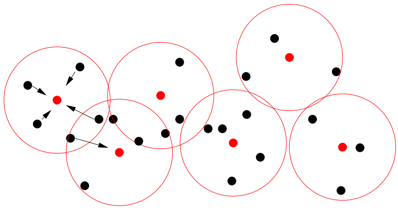
Protocol

1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
   - Every delegate reserves blocks of time steps for local use.
2. **Trigger:** delegates flood τ_1 and define tree, starting from sink.
3. **Collection:** slugs pass input value to chosen delegate.

Protocol

1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
   - Every delegate reserves blocks of time steps for local use.
2. **Trigger:** delegates flood τ_1 and define tree, starting from sink.
3. **Collection:** slugs pass input value to chosen delegate.

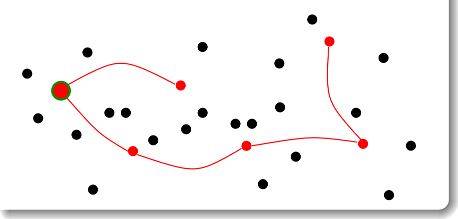
Windowed protocol



Protocol

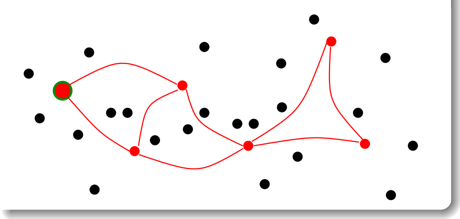
1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
   - Every delegate reserves blocks of time steps for local use.
2. **Trigger:** delegates flood τ_1 and define tree, starting from sink.
3. **Collection:** slugs pass input value to chosen delegate.
4. **Computation & Dissemination:** tree-based AND mass-distribution.

Tree-based



Aggregate at sink.

Mass distribution

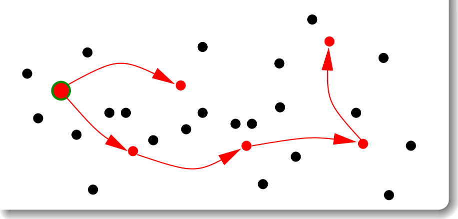


Iteratively share a fraction.

Protocol

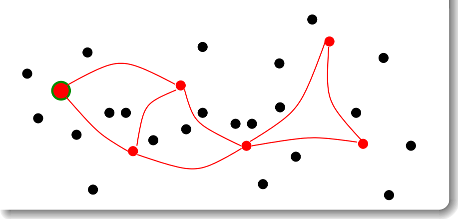
1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
   - Every delegate reserves blocks of time steps for local use.
2. **Trigger:** delegates flood τ_1 and define tree, starting from sink.
3. **Collection:** slugs pass input value to chosen delegate.
4. **Computation & Dissemination:** tree-based AND mass-distribution.

Tree-based



Aggregate at sink.

Mass distribution

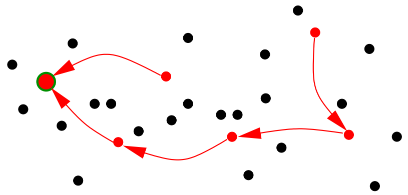


Iteratively share a fraction.

Protocol

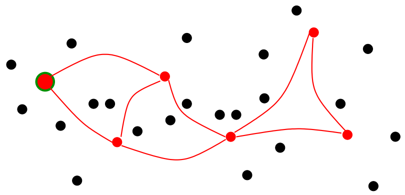
1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
   - Every delegate reserves blocks of time steps for local use.
2. **Trigger:** delegates flood τ_1 and define tree, starting from sink.
3. **Collection:** slugs pass input value to chosen delegate.
4. **Computation & Dissemination:** tree-based AND mass-distribution.

Tree-based



Aggregate at sink.

Mass distribution

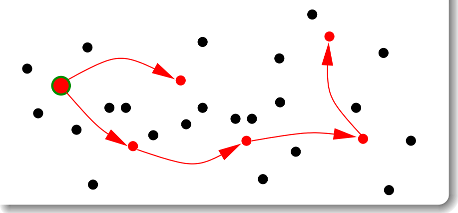


Iteratively share a fraction.

Protocol

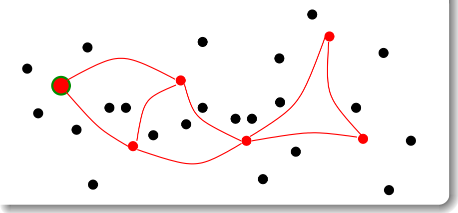
1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
   - Every delegate reserves blocks of time steps for local use.
2. **Trigger:** delegates flood τ_1 and define tree, starting from sink.
3. **Collection:** slugs pass input value to chosen delegate.
4. **Computation & Dissemination:** tree-based AND mass-distribution.

Tree-based



Aggregate at sink.

Mass distribution

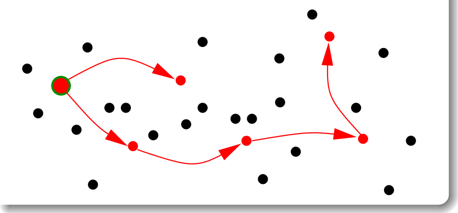


Iteratively share a fraction.

Protocol

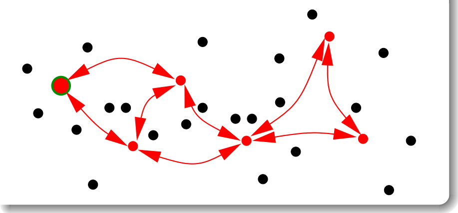
1. **Preprocessing:**
   - Partition nodes in *delegates* and *slugs*.
   - Every delegate reserves blocks of time steps for local use.
2. **Trigger:** delegates flood τ_1 and define tree, starting from sink.
3. **Collection:** slugs pass input value to chosen delegate.
4. **Computation & Dissemination:** tree-based AND mass-distribution.

Tree-based



Aggregate at sink.

Mass distribution



Iteratively share a fraction.

Bounds

1. **Preprocessing:**
   - MIS(αr). W.h.p. node i is in the partition within

$$O(\log^2 n) \text{ steps [MW'05].}$$

   - Coloring(r). W.h.p. delegate i reserves a block within

$$O(\log n) \text{ steps [FCM'07].}$$

2. **Trigger:** BFS(sink). Node i receives τ_1 within

$$O(D) \text{ steps.}$$

3. **Collection:** W.h.p. delegate i receives all slug values within

$$O(\Delta + \log^2 n) \text{ steps.}$$

Up to here, $O(D + \Delta)$ steps w.h.p.

Bounds

4. **Computation & Dissemination:**
   - tree-based: w.h.p. node i holds final value in
     $$O(D + f \log^2 n) \text{ steps.}$$
   - mass-distribution: w.h.p. node i holds final value in
     $$O\left(\frac{f - \log \varepsilon + \log(\nu_{max}/\nu_{min})}{\Phi_{min}^2}\right) \text{ steps.}$$

   $\Phi_{min} = \min_{k \in \{0,1,\ldots,f\}} \Phi_k$.
   Φ_k: conductance of underlying graph after kth failure.
   ε: relative error

   Adding $O(D + \Delta)$ to these bounds...

Bounds

Overall time efficiency

Theorem

$\exists \kappa_1, \kappa_2 > 0$ such that, if $T \geq \kappa_2 \log^2 n$, w.h.p., within

$$O(\Delta + D + f \log^2 n) \text{ time steps after } \tau_1 - \kappa_1(D + \log^2 n),$$

all nodes hold the same value in the range

$$\left[\frac{\overline{\nu}|V'| - f\nu_{min}}{|V'| - f}, \frac{\overline{\nu}|V'| - f\nu_{max}}{|V'| - f}\right].$$

Optimal if $f \in o(n^c)$ for any constant c.

Bounds

Overall time efficiency

Theorem

$\exists \kappa_1, \kappa_2 > 0$ such that, if $T < \kappa_2 \log^2 n$, w.h.p., within

$$O\left(\Delta + D + \frac{f - \log \varepsilon + \log \frac{\nu_{max}}{\nu_{min}}}{\Phi_{min}^2}\right) \text{ time steps after } \tau_1 - \kappa_1(D + \log^2 n),$$

all nodes have converged to a value in the range

$$[\nu_{max}, \nu_{min}]$$

with relative error $0 < \varepsilon < 1$.

Conclusions

- Combined algorithm is early stopping.
  - Non-frequent failures
    $\longrightarrow$ tree-based returns result fast and aborts mass-distribution.
  - Frequent failures
    $\longrightarrow$ mass-distribution returns at least an approximation later.
- All analyses include all communication costs.
- First optimal early-stopping for aggregation.

Open problems

- Only one radius.
- Geographic average.
- Other hierarchical topologies.
- Relax some restrictions.
- Mobile.

Thank you