Self-stabilization and Sensor Networks

Franck Petit / Sébastien Tixeuil

UPMC
Firstname.Lastname@lip6.fr
Outline

Sensor Networks and Self-stabilization
  Model(s)
  Cached Sensornet
  Self-stabilizing Unison

TDMA
  Motivation
  Algorithm stack

Clustering
  Density
  Self-stabilizing Clustering

Conclusion
Sensor Networks

- processor + sensors + radio
- 2 AA batteries, on/off switch
- 3 LEDs for debugging
Sensor Networks

While (batteries supply power)

- Collect, aggregate and reduce data
- log into memory

In spite of numerous fault modes

- Permanent sensor failures, node failures
- restarts, radio failures
- transient faults, reconfigurations
Distributed Systems

**Definition (Classical System, *a.k.a.* Non-stabilizing)**

Starting from a *particular* initial configuration, the system *immediately* exhibits correct behavior.

**Definition (Self-stabilizing System)**

Starting from *any* initial configuration, the system *eventually* reaches a configuration from with its behavior is correct.
Distributed Systems

Definition (Classical System, a.k.a. Non-stabilizing)
Starting from a particular initial configuration, the system immediately exhibits correct behavior.

Definition (Self-stabilizing System)
Starting from any initial configuration, the system eventually reaches a configuration from with its behavior is correct.

- Self-stabilization permits to recover from transient failures
Self-stabilization

Configurations

“Correct”

Stabilization Time
Complexity Criteria

Maximize useful lifetime of system

- “maximise useful”: correct quickly from illegitimate state
  - Self-stabilization, scalability
- “maximise lifetime”: use minimal energy to preserve batteries
  - local vs. global preserving
System Specifics

- only one radio frequency
- no collision detect
- access technique: CSMA/CA
- use CRC to detect collision
- no directional send/receive
- msg. are small (30 bytes)
- radio range about 1 meter
- number of neighbors < 10
- could be large number of nodes (perhaps > 100000)
- unique node IDs (probably)
- cost a few ¥ (someday)
- slow processor (4 MHz)
- limited memory (4 KB RAM)
- item nodes have real-time clocks \(\equiv\) drift between 1 msec and 100 msec per second
- several power modes available
The Model(s)
The Model(s)
The Model(s)
The Model(s)
The Model(s)

Self-stabilizing model
- Read neighborhood state,
- compute and update local state

Sensor Network model
- Read local state,
- compute and broadcast to neighborhood
- Collisions may appear
Self-stabilization in Sensor Networks

Transform (i.e. Simulate) the self-stabilizing model into the sensor networks model

- Pros: reuse existing SS algorithms
- Cons: potentially inefficient, overhead

Design self-stabilizing algorithms for the sensor networks model

- Pros: potentially efficient
- Cons: ignore previous SS work
Self-stabilization in Sensor Networks

Transform (i.e. Simulate) the self-stabilizing model into the sensor networks model

- Pros: reuse existing SS algorithms
- Cons: potentially inefficient, overhead
- [Herman 03] Cached Sensornet Transform

Design self-stabilizing algorithms for the sensor networks model

- Pros: potentially efficient
- Cons: ignore previous SS work
Self-stabilization in Sensor Networks

Transform (i.e. Simulate) the self-stabilizing model into the sensor networks model

- Pros: reuse existing SS algorithms
- Cons: potentially inefficient, overhead
- \[ \text{Cached Sensornet Transform} \]

Design self-stabilizing algorithms for the sensor networks model

- Pros: potentially efficient
- Cons: ignore previous SS work
- \[ \text{Unison with collisions} \]
Cached Sensornet Transform

Basic Algorithm

- Each node $p$ has a variable $v_p$
- Each neighbor $q$ of $p$ has a variable $c_qv_p$
  - $c_qv_p$ is the cached value of $v_p$ at $q$
- Whenever $p$ assigns $v_p$, $p$ also broadcasts the new value to the neighborhood
- Whenever a neighbor $q$ of $p$ receives $v_p$, $q$ updates $c_qv_p$ accordingly
Cached Sensornet Transform

Definition (Cache coherence)
For all neighbors $p$ and $q$, $c_q v_p = v_p$

Lemma (Closure)
If started from a cache coherent state, and without collisions, the self-stabilizing model is simulated by replacing all occurrences of $c_q v_p$ by $v_p$
Lemma (Closure)

If started from a cache coherent state, and without collisions, the self-stabilizing model is simulated by replacing all occurrences of $c_q v_p$ by $v_p$
Example

Lemma (Closure)

If started from a cache coherent state, and without collisions, the self-stabilizing model is simulated by replacing all occurrences of $c_q v_p$ by $v_p$.
Lemma (Closure)

If started from a cache coherent state, and without collisions, the self-stabilizing model is simulated by replacing all occurrences of $c_q v_p$ by $v_p$. 
Cached Sensornet Transform

Periodic retransmit

- Each node $p$ periodically broadcasts $v_p$ to its neighborhood

Lemma (Convergence)

If started from an arbitrary state, and without collisions, a cache coherent state is eventually reached
**Lemma (Convergence)**

*If started from an arbitrary state, and without collisions, a cache coherent state is eventually reached*
Lemma (Convergence)

If started from an arbitrary state, and without collisions, a cache coherent state is eventually reached.
Example

Lemma (Convergence)

If started from an arbitrary state, and without collisions, a cache coherent state is eventually reached.
Lemma (Convergence)

If started from an arbitrary state, and without collisions, a cache coherent state is eventually reached.
Lemma (Convergence)

If started from an arbitrary state, and without collisions, a cache coherent state is eventually reached.
Cached Sensornet Transform

Message Corruption

- Each neighbor $q$ of $p$ has a Boolean variable $b_{q}v_{p}$
- If $q$ receives $v_{p}$ correctly, $b_{q}v_{p}$ becomes true
- $G \rightarrow A$ becomes
  for all neighbors $q$ of $p$, $b_{p}v_{q}$ and $G \rightarrow A$; for all neighbors $q$ of $p$, $b_{p}v_{q}$ becomes false
Example

- If \( q \) receives \( v_p \) correctly, \( b_q v_p \) becomes true
- \( G \rightarrow A \) becomes
  for all neighbors \( q \) of \( p \), \( b_p v_q \) and \( G \rightarrow A \); for all neighbors \( q \) of \( p \), \( b_p v_q \) becomes false
Example

- If $q$ receives $v_p$ correctly, $b_q v_p$ becomes true
- $G \rightarrow A$ becomes for all neighbors $q$ of $p$, $b_p v_q$ and $G \rightarrow A$; for all neighbors $q$ of $p$, $b_p v_q$ becomes false
Example

- If \( q \) receives \( v_p \) correctly, \( b_q v_p \) becomes true
- \( G \rightarrow A \) becomes

  for all neighbors \( q \) of \( p \), \( b_p v_q \) and \( G \rightarrow A \); for all neighbors \( q \) of \( p \), \( b_p v_q \) becomes false
Example

- If \( q \) receives \( v_p \) correctly, \( b_q v_p \) becomes true
- \( G \rightarrow A \) becomes for all neighbors \( q \) of \( p \), \( b_p v_q \) and \( G \rightarrow A \); for all neighbors \( q \) of \( p \), \( b_p v_q \) becomes false
Example

- If \( q \) receives \( v_p \) correctly, \( b_q v_p \) becomes true
- \( G \rightarrow A \) becomes 
  for all neighbors \( q \) of \( p \), \( b_p v_q \) and \( G \rightarrow A \); for all neighbors \( q \) of \( p \), \( b_p v_q \) becomes false
Example

- If $q$ receives $v_p$ correctly, $b_q v_p$ becomes true.
- $G \rightarrow A$ becomes
  for all neighbors $q$ of $p$, $b_p v_q$ and $G \rightarrow A$;
  for all neighbors $q$ of $p$, $b_p v_q$ becomes false.
Example

- If $q$ receives $v_p$ correctly, $b_q v_p$ becomes true
- $G \rightarrow A$ becomes
  for all neighbors $q$ of $p$, $b_p v_q$ and $G \rightarrow A$; for all neighbors $q$ of $p$, $b_p v_q$ becomes false
Example

- If \( q \) receives \( v_p \) correctly, \( b_q v_p \) becomes true
- \( G \rightarrow A \) becomes
  for all neighbors \( q \) of \( p \), \( b_p v_q \) and \( G \rightarrow A \); for all neighbors \( q \) of \( p \), \( b_p v_q \) becomes false
Cached Sensornet Transform

Periodic Retransmit

Message Corruption

Lemma (Self-stabilization)
If started from an arbitrary state, the self-stabilizing model is eventually simulated
Self-stabilizing Unison

Specification

- Each node $p$ has a clock variable $v_p$
- For every neighbors $p$ and $q$, $|v_p - v_q| \leq 1$
Self-stabilizing Unison

Specification

- Each node $p$ has a clock variable $v_p$
- For every neighbors $p$ and $q$, $|v_p - v_q| \leq 1$

Self-stabilizing Unison

- for every neighbor $q$, $v_q \geq v_p \rightarrow v_p := v_p + 1$
Example

Self-stabilizing Unison

- for every neighbor $q$, $v_q \geq v_p \rightarrow v_p := v_p + 1$
Example

Self-stabilizing Unison

- for every neighbor \( q \), \( v_q \geq v_p \rightarrow v_p := v_p + 1 \)
Example

Self-stabilizing Unison

- for every neighbor $q$, $v_q \geq v_p \rightarrow v_p := v_p + 1$
Example

Self-stabilizing Unison

- for every neighbor $q$, $v_q \geq v_p \rightarrow v_p := v_p + 1$

Nodes:
- Grey: non activatable
- Pink: activatable
- Green: legitimate
Example

Self-stabilizing Unison

- for every neighbor $q$, $v_q \geq v_p \rightarrow v_p := v_p + 1$
Example

Self-stabilizing Unison

- for every neighbor $q$, $v_q \geq v_p \rightarrow v_p := v_p + 1$
Example

Self-stabilizing Unison

- for every neighbor $q$, $v_q \geq v_p \rightarrow v_p := v_p + 1$
Example

Self-stabilizing Unison

- for every neighbor \( q \), \( v_q \geq v_p \rightarrow v_p := v_p + 1 \)
Self-stabilizing Unison

- For every neighbor $q$, $v_q \geq v_p \rightarrow v_p := v_p + 1$
Example

Self-stabilizing Unison

- for every neighbor $q$, $v_q \geq v_p \rightarrow v_p := v_p + 1$
Example

Self-stabilizing Unison

- for every neighbor $q$, $v_q \geq v_p \rightarrow v_p := v_p + 1$
Example

Self-stabilizing Unison

- for every neighbor $q$, $v_q \geq v_p \rightarrow v_p := v_p + 1$
Example

Self-stabilizing Unison

- for every neighbor $q$, $v_q \geq v_p \rightarrow v_p := v_p + 1$
Example

Self-stabilizing Unison

- for every neighbor $q$, $v_q \geq v_p \rightarrow v_p := v_p + 1$
Unison with Collisions

Specification

- Each node $p$ has a clock variable $v_p$
- For every neighbors $p$ and $q$, $|v_p - v_q| \leq 1$

Self-stabilizing Unison

- for every neighbor $q$, $v_q \geq v_p \rightarrow v_p := v_p + 1$
Unison with Collisions

Specification

- Each node $p$ has a clock variable $v_p$
- For every neighbors $p$ and $q$, $|v_p - v_q| \leq 1$

Self-stabilizing Unison with Collisions

- for every neighbor $q$, $c_p v_q \geq v_p \rightarrow v_p := v_p + 1$
Unison with Collisions

Specification

► Each node $p$ has a clock variable $v_p$
► For every neighbors $p$ and $q$, $|v_p - v_q| \leq 1$

Self-stabilizing Unison with Collisions

► for every neighbor $q$, $c_p v_q \geq v_p \rightarrow v_p := v_p + 1$
► Only correctly received messages update cached variables
Example

- **non activatable**
- **activatable**
- **legitimate**

- **lower than value**
- **strictly greater**

Diagram:

1  2  0  2  4  1  2
Example

- non activatable
- activatable
- legitimate
- lower than value
- strictly greater

3  2  0  2  1  4

Sensor Networks and Self-stabilization
TDMA
Clustering
Conclusion
Example

- non activatable
- activatable
- legitimate
- lower than value
- strictly greater

```
3  2  0  2  4  4
```

- 3
- 2
- 0
- 2
- 4
- 4
Example

- gray: non activatable
- pink: activatable
- green: legitimate
- □: lower than value
- □: strictly greater

Diagram:

- Node 3: 2
- Node 2: 0
- Node 4: 2
- Node 4: 4

Connections:
- 3 → 2
- 2 → 4
- 4 ← 2
**Example**

- non activatable
- activatable
- legitimate
- lower than value
- strictly greater

![Diagram](image)
Example

- **Non activatable**
- **Activatable**
- **Legitimate**
- **Lower than value**
- **Strictly greater**

Diagram with nodes labeled 3, 3, 3, 4, 4, connected with arrows and colored symbols.
Example

- non activatable
- activatable
- legitimate
- lower than value
- strictly greater
Unison with Collisions

Cache coherence weakening

- For every neighbors $p$ and $q$, $c_p v_q \leq v_q$

Self-stabilizing Unison with collisions

- Unison and Weak cache coherence are preserved by program executions
- Unison and Weak cache coherence eventually hold
- Some extra work is expected to get bounded clock values
Self-stabilization in Sensor Networks

Transform (i.e. Simulate) the self-stabilizing model into the sensor networks model

- [Herman 03] Cached Sensornet Transform
- Overhead is not upper bounded

Design self-stabilizing algorithms for the sensor networks model

- [Herman 03] Unison with collisions
- Proof in the model is specific to the problem
# Outline

**Sensor Networks and Self-stabilization**
- Model(s)
- Cached Sensornet
- Self-stabilizing Unison

**TDMA**
- Motivation
- Algorithm stack

**Clustering**
- Density
- Self-stabilizing Clustering

**Conclusion**
Towards an Intermediate Model

An atomic step at a node

- Compute new state, write new state at all neighbors (no collision)

Hypothesis

- Global clock, unique IDs

Solution

- TDMA to avoid collisions
Towards an Intermediate Model

Solution

- TDMA to avoid collisions
- assume synchronised, real-time clocks (to enable TDMA slotted time)
- but TDMA implemented using CSMA/CA as basic, underlying model
TDMA Scheduling

- Algorithm messages are transmitted during the "overhead" periods
- TDMA slot assignment is the output of our algorithm
Self-stabilizing TDMA for Sensors

- [Kulkarni, Arumugam 03] 2-D Grids
  - nodes are aware of their positions
  - Not suitable for dynamic/faulty networks
- [Herman, Tixeuil 04] General graphs of bounded degree
  - Randomized algorithm, self-stabilizing in expected $O(1)$ time, to assign TDMA slots
  - Solution is a protocol stack based on variable propagation, minimal coloring of $N^2$, MIS construction, and mapping colors $\leftrightarrow$ TDMA slots
Self-stabilizing TDMA for Sensors

- ▶ both are minimal,
- ▶ but second solution is better for time-slot assignment
Self-stabilizing TDMA for Sensors

- both are minimal,
- but second solution is better for time-slot assignment
Overview

$N^2$ minimal coloring $\rightarrow$ TDMA schedule

$N^3$ coloring + MIS $\rightarrow$ $N^2$ minimal coloring

$N^3$ coloring $\rightarrow$ MIS

Variables propagation $\rightarrow$ $N^3$ coloring

CSMA/CA $\rightarrow$ Variables propagation
CSMA/CA → Variables propagation

- Wait fixed delay
  - to process received messages, and update local variables
- Wait random delay
  - to allow Aloha-style analysis for probability of collisions among neighbors
- “Age” information to remove invalid data
Shared variables $\rightarrow N^3$ coloring

\[
\exists j \in N_i^3, \text{color}_j = \text{color}_i \rightarrow \text{color}_i := \text{random}(\Delta \setminus \{\text{color}_j | j \in N_i^3\})
\]

- Stabilizes in expected $O(1)$
- Output an ID-based DAG of constant height
$N^3$ coloring $\rightarrow$ MIS

[Ikeda, Kamei, Kakugawa 02]

No parent in MIS $\rightarrow$ join MIS
$N^3$ coloring $\rightarrow$ MIS

[Ikeda, Kamei, Kakugawa 02]

No parent in MIS $\rightarrow$ join MIS
$N^3$ coloring → MIS

[Ikeda, Kamei, Kakugawa 02]

No parent in MIS → join MIS
$N^3$ coloring → MIS

[Ikeda, Kamei, Kakugawa 02]

No parent in MIS → join MIS
$N^3$ coloring $\rightarrow$ MIS

[Ikeda, Kamei, Kakugawa 02]

No parent in MIS $\rightarrow$ join MIS
$N^3$ coloring $\rightarrow$ MIS

[Ikeda, Kamei, Kakugawa 02]

No parent in MIS $\rightarrow$ join MIS
$N^3$ coloring + MIS → Minimal $N^2$ coloring

MIS → send colors to dominated nodes
$N^3$ coloring + MIS $\rightarrow$ Minimal $N^2$ coloring

MIS $\rightarrow$ send colors to dominated nodes
\(N^3\) coloring + MIS $\rightarrow$ Minimal \(N^2\) coloring

MIS $\rightarrow$ send colors to dominated nodes
$N^3$ coloring + MIS $\rightarrow$ Minimal $N^2$ coloring

MIS $\rightarrow$ send colors to dominated nodes
$N^3$ coloring $+$ MIS $\rightarrow$ Minimal $N^2$ coloring

MIS $\rightarrow$ send colors to dominated nodes
Minimal $N^2$ coloring → TDMA Schedule
Minimal $N^2$ coloring $\rightarrow$ TDMA Schedule
Minimal $N^2$ coloring $\rightarrow$ TDMA Schedule
Minimal $N^2$ coloring $\rightarrow$ TDMA Schedule
Minimal $N^2$ coloring $\rightarrow$ TDMA Schedule
Minimal $N^2$ coloring $\rightarrow$ TDMA Schedule
Minimal $N^2$ coloring $\rightarrow$ TDMA Schedule
Minimal $N^2$ coloring $\rightarrow$ TDMA Schedule
Minimal $N^2$ coloring $\rightarrow$ TDMA Schedule
Minimal $N^2$ coloring → TDMA Schedule
Minimal $N^2$ coloring $→$ TDMA Schedule
Outline

Sensor Networks and Self-stabilization
  Model(s)
  Cached Sensornet
  Self-stabilizing Unison

TDMA
  Motivation
  Algorithm stack

Clustering
  Density
  Self-stabilizing Clustering

Conclusion
Motivation

Clusters for routing
MANET routing protocols are flat, thus not scalable
Cluster-heads have extra responsibility for the routing of message

Cluster-heads should be stable
Handle departures and removals Handle node mobility
Density

\[ \rho(u) = \frac{\{e = (v, w) \in E \mid w \in \{u\} \cup N_u \text{ and } v \in N_u\}}{|N_u|} \]
Density

\[ \rho(u) = \frac{|\{e = (v, w) \in E \mid w \in \{u\} \cup N_u \text{ and } v \in N_u\}|}{|N_u|} \]
Density

$$\rho(u) = \frac{|\{e = (v, w) \in E \mid w \in \{u\} \cup N_u \text{ and } v \in N_u\}|}{|N_u|}$$
Cluster Head Heuristics
Cluster Head Heuristics

- $a = 1.33$
- $b = 1.75$
- $c = 1$
- $d = 1.75$
- $e = 1.8$
- $f = 1.33$
- $g = 1.5$
- $h = 1$
- $i = 1$

Cluster Head Heuristics Diagram
Cluster Head Heuristics

- $a_{1.33}$
- $b_{1.75}$
- $c_{1}$
- $d_{1.75}$
- $e_{1.8}$
- $f_{1.33}$
- $g_{1.5}$
- $h_{1}$
- $i_{1}$
- $j_{1.5}$
Self-stabilizing Clustering

Basic Idea

- Identify $N$ and $N^2$
- Compute and broadcast density
- Attach to neighbor with higher density
- use identifiers to break ties
Self-stabilizing Clustering

Basic Idea

- Identify $N$ and $N^2$
- Compute and broadcast density
- Attach to neighbor with higher density
- use identifiers to break ties
- Can be $O(\text{Diameter})$ if graph is regular
Faster Self-stabilizing Clustering

Basic Idea

- Identify $N$ and $N^2$
- Compute and broadcast density
- Random $L(1,1)$ coloring with $\delta^2$ colors
  - This can be done in expected $O(1)$ time
- Attach to neighbor with higher density
- Use colors to break ties
Faster Self-stabilizing Clustering

Basic Idea

- Identify $N$ and $N^2$
- Compute and broadcast density
- Random $L(1,1)$ coloring with $\delta^2$ colors
  - This can be done in expected $O(1)$ time
- Attach to neighbor with higher density
- use colors to break ties
- Expected constant stabilization time
- Use lexicographic order (density, color)
Conclusion

- Self-stabilization is interesting for sensor networks
  - Known SS solutions should be implemented in sensor networks
- Sensor networks are interesting for self-stabilization
  - Simple devices
  - Small operating system
Conclusion

- Self-stabilization is interesting for sensor networks
  - Known SS solutions should be implemented in sensor networks
- Sensor networks are interesting for self-stabilization
  - Simple devices
  - Small operating system
- Energy constraints and collisions make things complicated