Firefly Synchronization with Asynchronous Wake-Up

Dan Alistarh, Alejandro Cornejo, Mohsen Ghaffari, Nancy Lynch
Fireflies synchronize
Research on Firefly Synch

• Early research
  – E.g., [Smith35], [Buck88]
  – [Peskin73], [KuramotoN87]

• Mirollo-Strogatz [MS90]
  – Dynamical system model for the phenomenon, explaining synchronization in a clique

• Sparked considerable research on applications
  – Clock synchronization in computer systems
    [LucarelliWang05, Gopal06, SimeoneS08, etc.]
Abstract model

- N nodes (fireflies) in a *connected* topology
  - wake up at arbitrary times
- Communicate through *beeps (pulses)*
  - *Binary* information
  - Only neighbors can “see” pulse
Synchronization

- **Synch**: exists *Global Synch Time (GST)*, period $T > 1$, and **offset o** such that, after GST:
  - Nodes beep at global time $t$ if $(t - o) \mod T = 0$
  - Don’t beep otherwise
Time models

• Nodes:
  – share a period $T$
  – beep once per period

• Node dynamics
  – either continuous (integrate-and-fire)
  – or discrete (averaging)

• Continuous time:
  – Characterized by a dynamical system
  – Fixed point: all nodes beep synchronously

• Discrete
  – Characterized by a system of equations
  – Sync: all nodes beep in the same time slot
Discrete time

• Time divided into discrete, *aligned* slots

• Each node $i$:
  
  – Wakes up at (global) time $w_i$
  
  – Beeps once in every period
    
    $t_i(k) = w_i + kT + \tau_i(k)$
  
  – Averages over its neighbors:
    
    $\tau_i(k) = \frac{1}{\Delta_i} \sum_j (t_j(k-1) - t_i(k-1))$
  
  – System: $t(k) = A \ t(k-1)$, where $A$ is a Laplacian

All this is well known, and seems to work fine.

However...
The problem

- The algorithm does not always converge!

$$t_i(k) = w_i + (k - 1) \cdot T + \frac{1}{\Delta} \sum_j (t_j(k - 1) - t_i(k - 1))$$

The problem:
The round structure is not respected under asynchronous wakeup!
The problem (2)

The system equation no longer holds!
...and in fact, the system does not converge.

Assuming synchronous wake-up not a solution, since then nodes are already synchronized.
Our project

There is a problem with “averaging” algorithms (even if initial offsets are less than $T$)

• Hint of a solution:
  – We give an averaging algorithm, under assumptions on system parameters
  – Simple non-averaging algorithm

• Interesting open questions
The Algorithm

Wake-up phase
• the adversary wakes up a subset of the nodes
• a node beeps as soon as it wakes up
• sets its next beep $T/2$ slots later

Convergence phase
• nodes then start *averaging* in each “round”
• average rounded *down*

Assumptions:
1. Each node wakes up its neighbors by beeping
2. $T \geq 4n$

- $O(D^2)$ rounds follows easily
- $O(D)$ rounds the right answer
O(D) Round Analysis

- **Claim 1**: Rounds are communication-closed.
- **Claim 2**: Neighbors are always at most one slot apart.
- For node $v$, round $k$, diameter $D$, define

$$F(v,k) = (1 + 1/D) \cdot \text{offset}(v, \text{root}) - \text{dist}(v, \text{root}) + k - 1$$

- **Claim 2**: For any $v$, either

  $$\text{offset}(v, \text{root}) = 0,$$  
  $$\text{or } F(v, k) \leq D + 1.$$  

- For $k > 2D + 2$, $\text{offset}(v, \text{root}) = 0$, so nodes are in sync.

*Averaging works in $O(DT)$ time units, given “gradient property” and $T \geq 4n$.*
A simpler algorithm

• Under the same assumptions, consider the trivial move-to-the-left (if you see something to your left) algorithm

• It also converges in $O(TD)$ time

The “gradient property” + $T \geq 4n$ trivialize the problem to some extent.
Open questions

• We gave a working averaging algorithm
• Two strong assumptions:
  – Nodes wake up on neighbor beep (gradient)
  – $T \geq 4n$ (consistent rounds)

How about asynchronous wakeup?

How do they do it?

Lower bounds?