Near-Optimal Radio Use For Wireless Network Synchronization

Milan Bradonjić, Eddie Kohler, Rafail Ostrovsky

LANL, UCLA

10th of July, 2009

Motivation

- Consider sensor network: tiny, inexpensive embedded computers
  - run complex software
  - sense environmental phenomena
  - communicate over wireless channels
- Typically, listening or transmitting $\approx 100\times$ expensive than idle CPU or radio switched off.
- Radio use $\gg$ computation cost

Goals

- System designers: power down radios as much as possible
- Successful communication requires synchronization of radio devices

Previous Literature

Extensively studied in practice - deployment and clock synchronization of wireless sensor networks:

- McGlynn and Borbash 2001,
- Tseng, Hsu and Hseih 2003,
- PalChaudhuri and Johnson 2004,
- Moscibroda, Von Rickenbach and Wattenhofer 2006,
- Sundararaman, Buy and Kshemkalyani 2005.

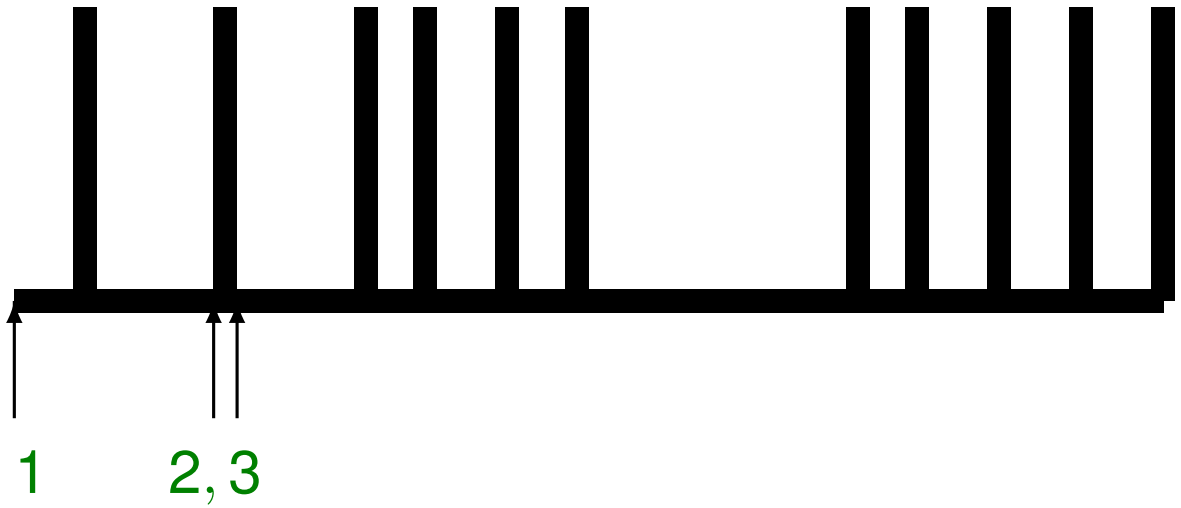
Two processors case

Warm-up:

- assume (for now) two processors
- clocks are at most n steps apart
- processors should try to synchronize within $4n$ steps

Picture Interpretation

1. CPU and CLOCK start
2. radio power ON
3. radio power OFF



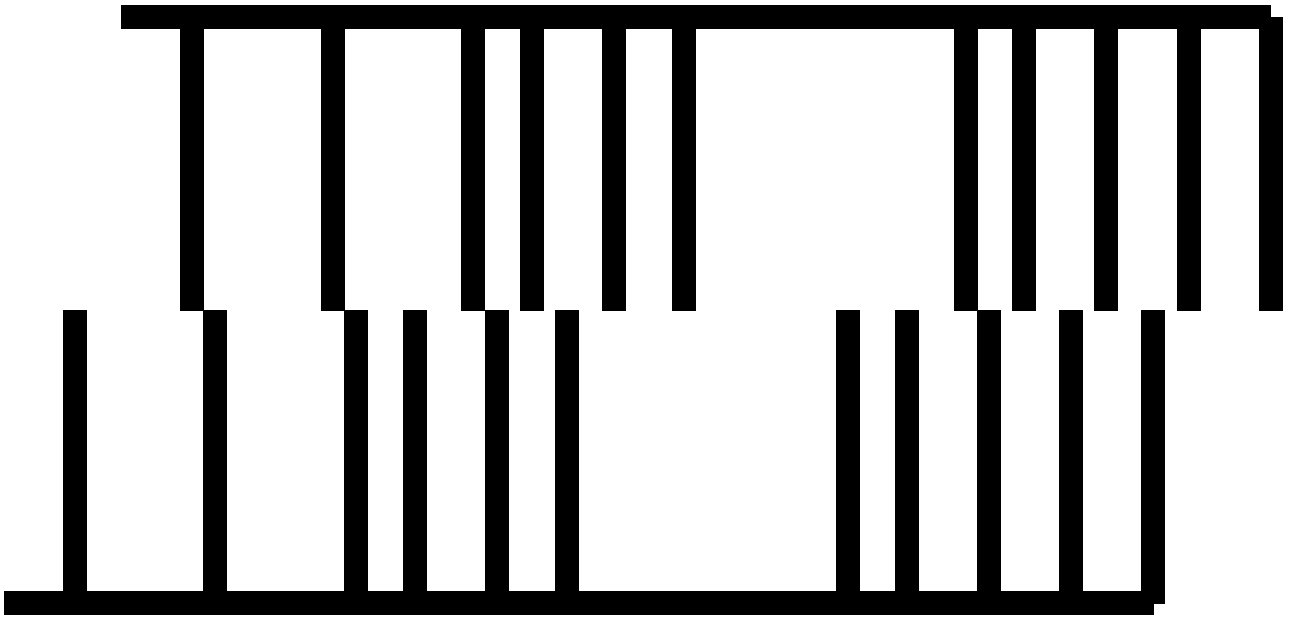
“Good Solution”

- Which “awake” solutions are good?
- Those that for any shift between 1 and n will always meet (remember that strings are of length at least $4n$).

“Good Solution”

- Which “awake” solutions are good?
- Those that for any shift between 1 and n will always meet (remember that strings are of length at least $4n$).

“bad solution” for 2 strings: shift=10



Density of Strings

- We discretize time to short intervals: put 1 when radio is ON, put 0 when radio is OFF.
- Clearly, if we set all bits at positions 1 to n to one, then **all** right shifts from 1 to n will result in meeting of two strings.
- Q What is the smallest density needed such that, all right shifts from 1 to n result in meeting of two strings?

Density of Strings

- We discretize time to short intervals: put 1 when radio is ON, put 0 when radio is OFF.
- Clearly, if we set all bits at positions 1 to n to one, then **all** right shifts from 1 to n will result in meeting of two strings.

Q What is the smallest density needed such that, all right shifts from 1 to n result in meeting of two strings?

Density of Strings

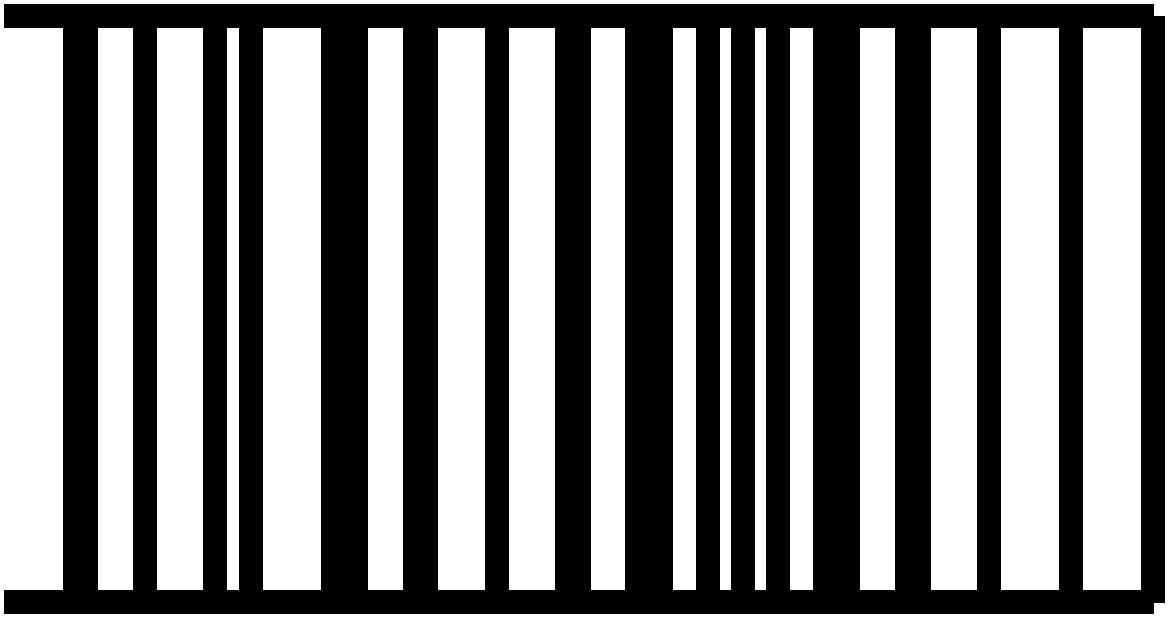
- We discretize time to short intervals: put 1 when radio is ON, put 0 when radio is OFF.
- Clearly, if we set all bits at positions 1 to n to one, then **all** right shifts from 1 to n will result in meeting of two strings.

Q What is the smallest density needed such that, all right shifts from 1 to n result in meeting of two strings?

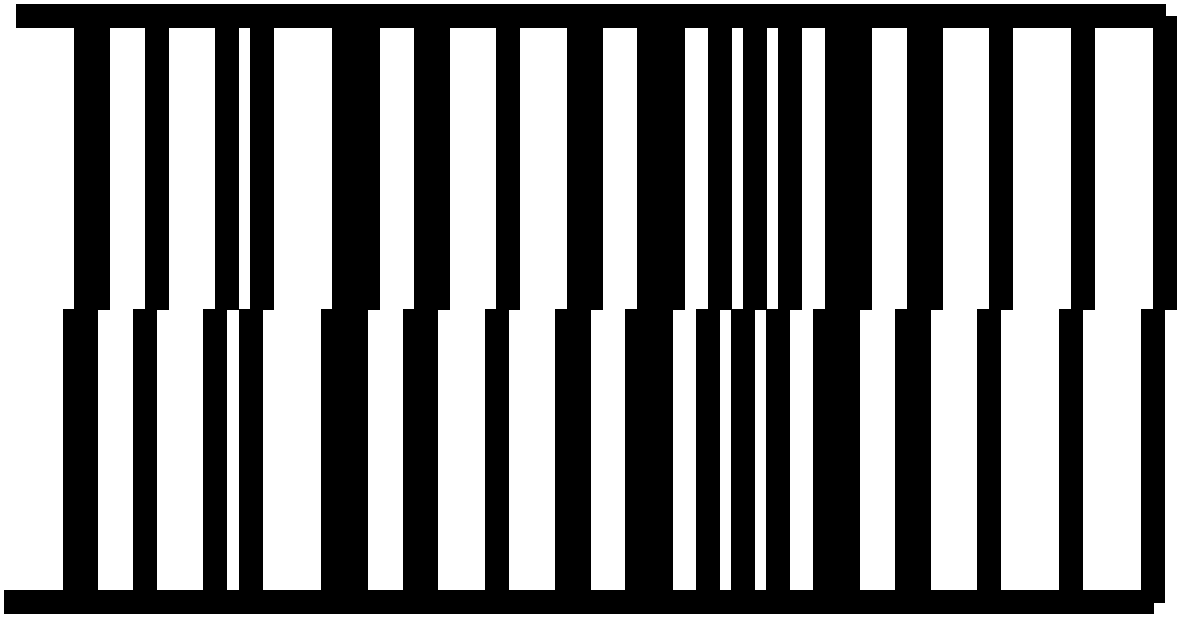
Example of a "Good Solution"

- For $n = 36$ we now consider two identical strings of the length $2n + 4\sqrt{n} + 2 = 98$.
- Then for any of $1, \ldots, 36$ right shifts, the following two strings meet.

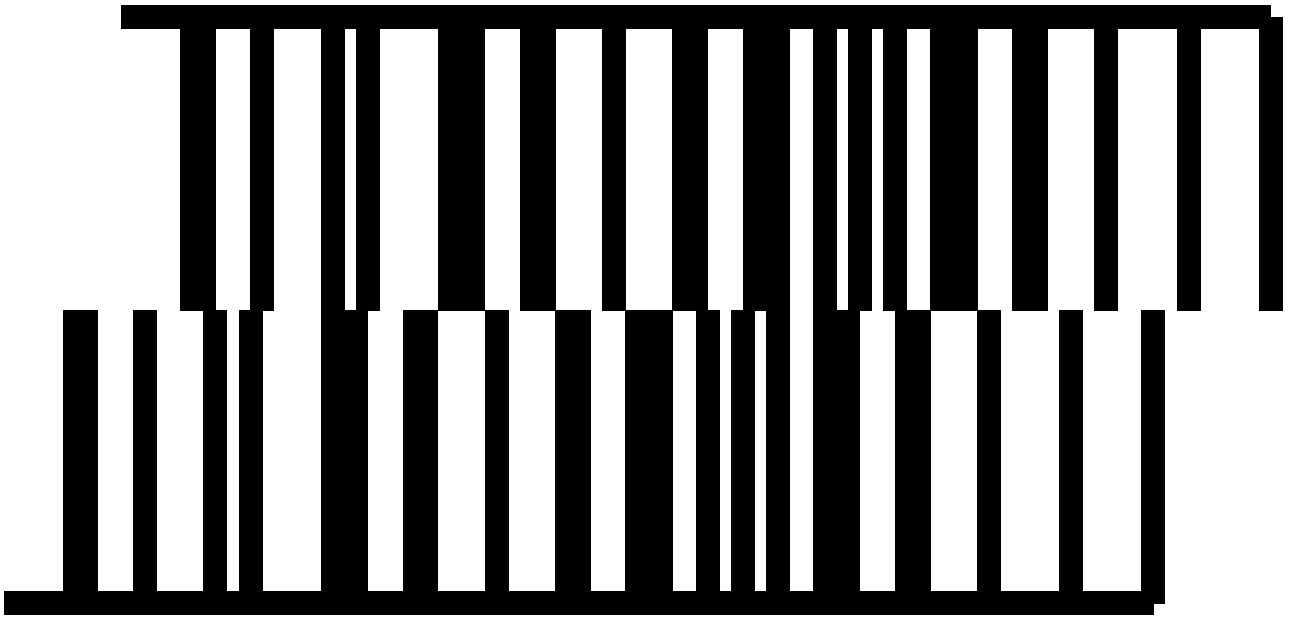
“good solution” for 2 strings: shift=0



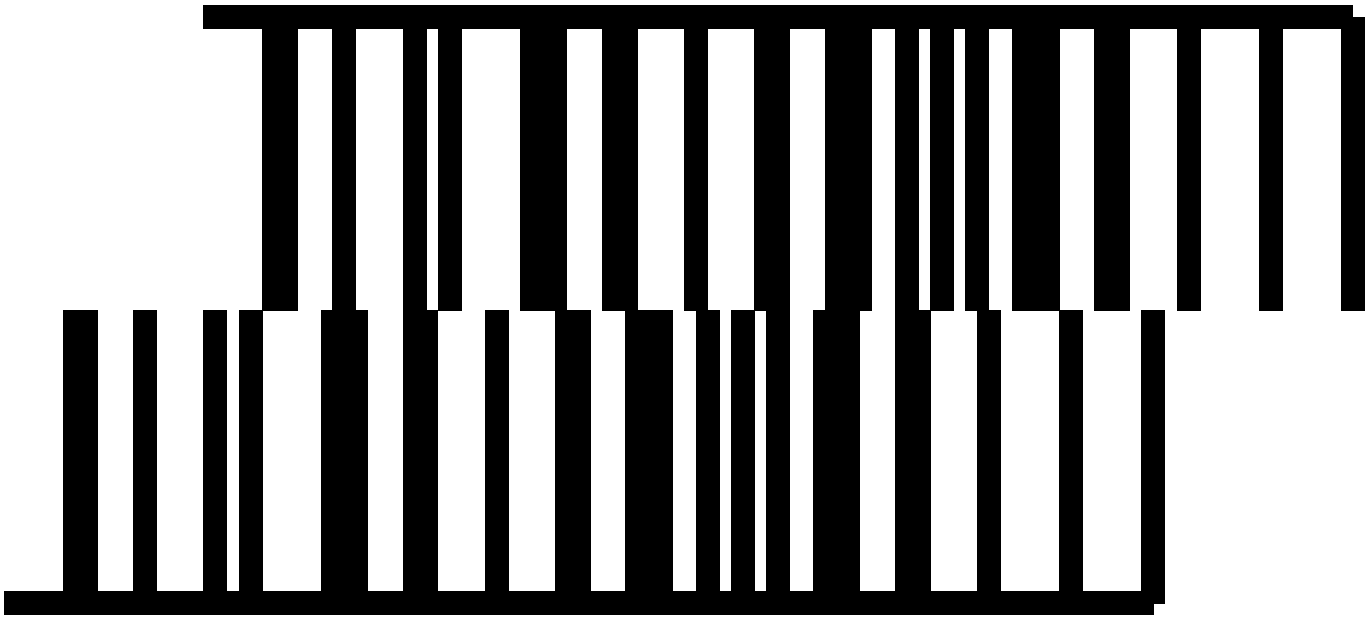
“good solution” for 2 strings: shift=1



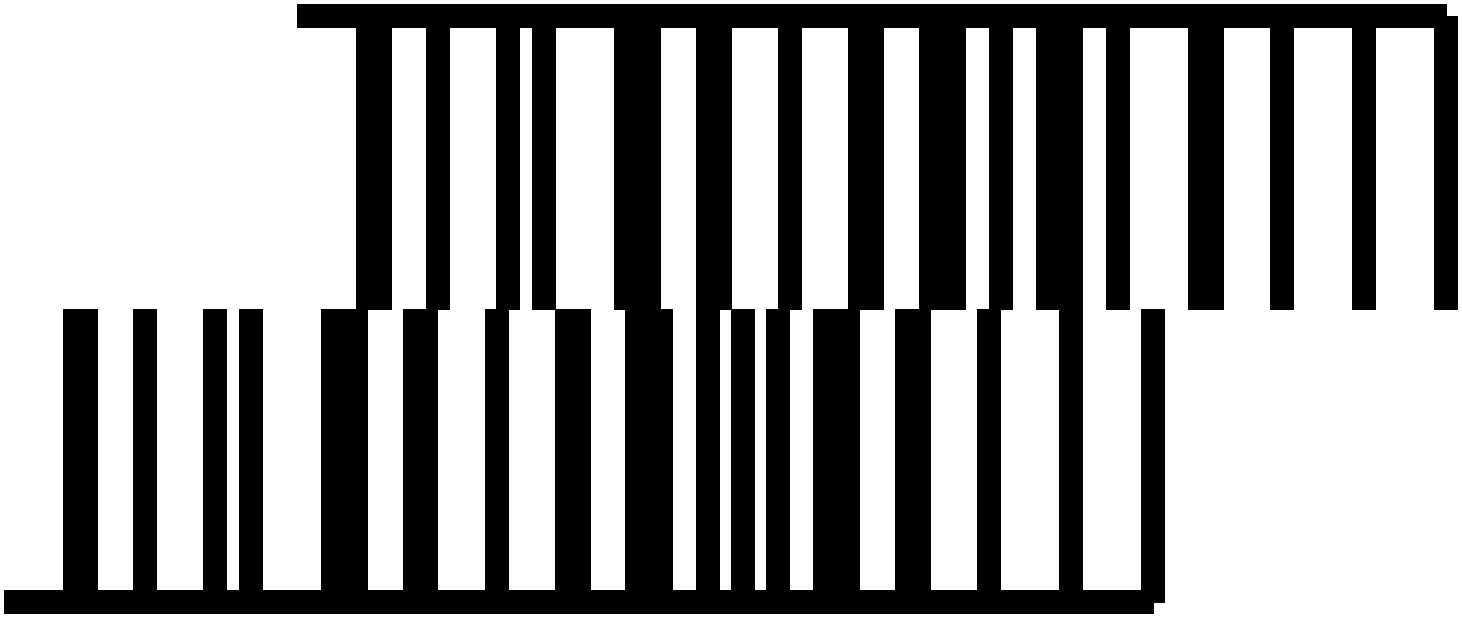
“good solution” for 2 strings: shift=10



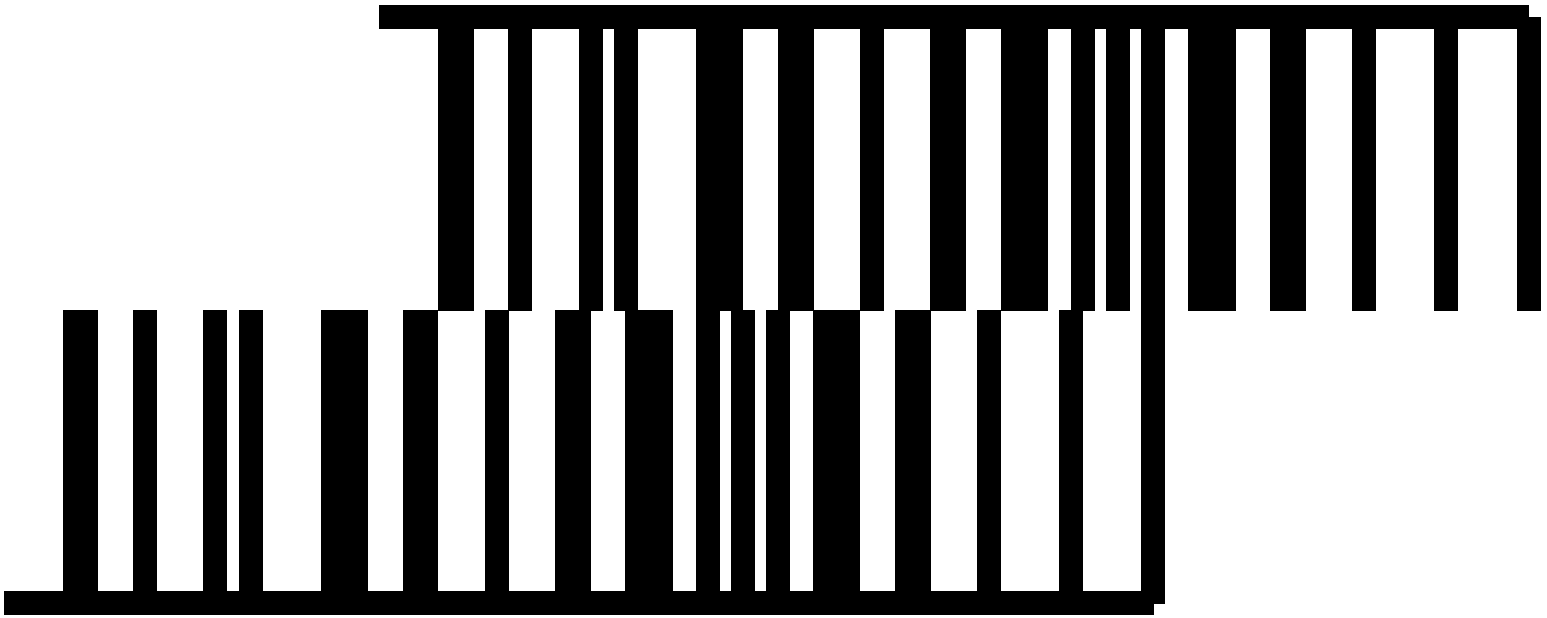
“good solution” for 2 strings: shift=17



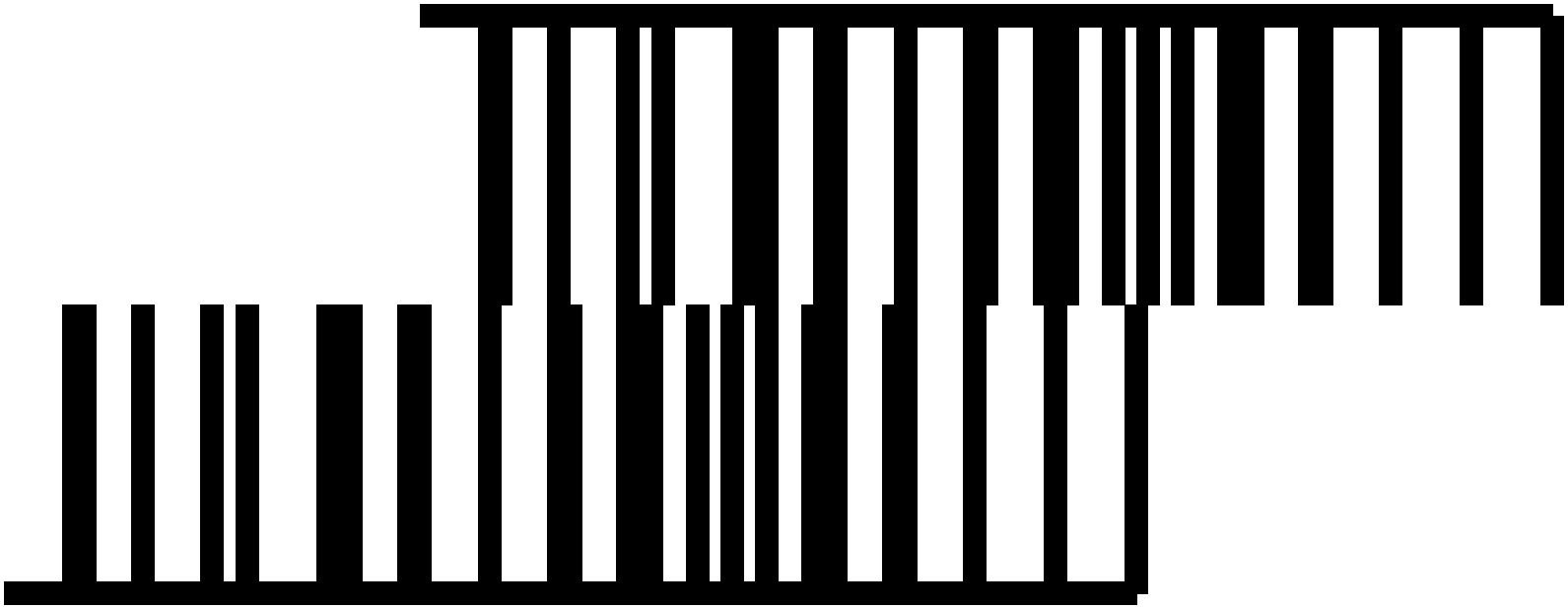
“good solution” for 2 strings: shift=25



“good solution” for 2 strings: shift=32



“good solution” for 2 strings: shift=36



- We explain how to derive the good solution, and do so deterministically.
- First, let us define the model.

Model and Problem Statement

- m radio devices
- Each device starts at an arbitrary time $\in \{0, 1, 2, \ldots, n\}$
- n is maximal initial clocks' difference
- At each time unit, device can be "awake" or "sleeping"
- **Goal:** Adjust clocks for all m devices, under objective to minimize radio use per processor, (i.e. the total time of radio being awake)
- Extension 1: Interference – if exactly two awake processors, then they can communicate with each other
- Extension 2: Model can capture different clocks' speeds and interference

Model and Problem Statement

- m radio devices
- Each device starts at an arbitrary time $\in \{0, 1, 2, \ldots, n\}$
- n is maximal initial clocks' difference
- At each time unit, device can be "awake" or "sleeping"
- **Goal:** Adjust clocks for all m devices, under objective to minimize radio use per processor, (i.e. the total time of radio being awake)
- Extension 1: Interference – if exactly two awake processors, then they can communicate with each other
- Extension 2: Model can capture different clocks' speeds and interference

Model and Problem Statement

- m radio devices
- Each device starts at an arbitrary time $\in \{0, 1, 2, \ldots, n\}$
- n is maximal initial clocks' difference
- At each time unit, device can be "awake" or "sleeping"
- **Goal:** Adjust clocks for all m devices, under objective to minimize radio use per processor, (i.e. the total time of radio being awake)
- Extension 1: Interference – if exactly two awake processors, then they can communicate with each other
- Extension 2: Model can capture different clocks' speeds and interference

Main Results

Our bounds on the optimal radio use, per processor, in order to synchronize the network:

- Two processors, optimal use:
  - $\Omega(\sqrt{n})$ deterministic lower bound
  - matching deterministic $O(\sqrt{n})$ upper bound
- Arbitrary $m = n^{\beta}$ processors:
  - $\Omega\left(n^{\frac{1-\beta}{2}}\right)$ the lower bound for any deterministic protocol
  - nearly-matching $O\left(n^{\frac{1-\beta}{2}} \cdot \text{poly-log}(n)\right)$ for randomized protocol, whp
- If m is not known:
  - We can repeat 'the previous bullet' $O(\log n)$ times, and estimate the value of m.

Main Results

Our bounds on the optimal radio use, per processor, in order to synchronize the network:

- Two processors, optimal use:
  - $\Omega(\sqrt{n})$ deterministic lower bound
  - matching deterministic $O(\sqrt{n})$ upper bound
- Arbitrary $m = n^{\beta}$ processors:
  - $\Omega\left(n^{\frac{1-\beta}{2}}\right)$ the lower bound for any deterministic protocol
  - nearly-matching $O\left(n^{\frac{1-\beta}{2}} \cdot \text{poly-log}(n)\right)$ for randomized protocol, whp
- If m is not known:
  - We can repeat 'the previous bullet' $O(\log n)$ times, and estimate the value of m.

Main Results

Our bounds on the optimal radio use, per processor, in order to synchronize the network:

- Two processors, optimal use:
  - $\Omega(\sqrt{n})$ deterministic lower bound
  - matching deterministic $O(\sqrt{n})$ upper bound
- Arbitrary $m = n^{\beta}$ processors:
  - $\Omega\left(n^{\frac{1-\beta}{2}}\right)$ the lower bound for any deterministic protocol
  - nearly-matching $O\left(n^{\frac{1-\beta}{2}} \cdot \text{poly-log}(n)\right)$ for randomized protocol, whp
- If m is not known:
  - We can repeat 'the previous bullet' $O(\log n)$ times, and estimate the value of m.

Main Results

Our bounds on the optimal radio use, per processor, in order to synchronize the network:

- Two processors, optimal use:
  - $\Omega(\sqrt{n})$ deterministic lower bound
  - matching deterministic $O(\sqrt{n})$ upper bound
- Arbitrary $m = n^{\beta}$ processors:
  - $\Omega\left(n^{\frac{1-\beta}{2}}\right)$ the lower bound for any deterministic protocol
  - nearly-matching $O\left(n^{\frac{1-\beta}{2}} \cdot \text{poly-log}(n)\right)$ for randomized protocol, whp
- If m is not known:
  - We can repeat 'the previous bullet' $O(\log n)$ times, and estimate the value of m.

Lower Bound

The density of strings is ‘small’ $\Rightarrow$ there is a shift such that the strings do not ‘meet’.

Lemma (Two Non-Colliding Strings)

For any absolute constant $C \geq 1/\sqrt{2}$, and for every L-bit string with $\ell \leq \sqrt{L/C}$ ones, there is at least one shift within $L/(2C^2)$, such that the string and its shifted copy do not meet.

Lower Bound

The density of strings is ‘small’ $\Rightarrow$ there is a shift such that the strings do not ‘meet’.

Lemma (Two Non-Colliding Strings)

For any absolute constant $C \geq 1/\sqrt{2}$, and for every L-bit string with $\ell \leq \sqrt{L/C}$ ones, there is at least one shift within $L/(2C^2)$, such that the string and its shifted copy do not meet.

Proof Outline

- *Consider an L-bit string, with ℓ ones.*
- *Inspect the set of differences among the positions of these ℓ ones in the string.*
- *There are $\leq L$ of these differences (because of the density of ones in the string).*
- *Choose an integer from $\{1, \ldots, L\}$ not in the set of differences and shift the string for that number to the right.*
- *Now, the strings do not meet!*

Matching Upper Bound for Two Processors

We now show the upper bound and give the deterministic algorithm for two devices.

Matching Upper Bound for Two Processors

Theorem

For any n, there exists a string of length $W = 2n + 4\sqrt{n} + 2$ with at most $4\sqrt{n} + 4$ ones such that this string will overlap itself for all shifts from 1 to n.

Proof Outline

- *Set the bits at positions: $(i\sqrt{n} + i)$ and $(i\sqrt{n})$ to 1, for $i \in \{1, \ldots, \lfloor 2\sqrt{n} + 2 \rfloor\}$*
- *Set the remaining bits to 0*
- *We show that for any shift from 0 to n, two strings of the length W "meet" with probability 1*

Matching Upper Bound for Two Processors

Theorem

For any n, there exists a string of length $W = 2n + 4\sqrt{n} + 2$ with at most $4\sqrt{n} + 4$ ones such that this string will overlap itself for all shifts from 1 to n.

Proof Outline

- *Set the bits at positions: $(i\sqrt{n} + i)$ and $(i\sqrt{n})$ to 1, for $i \in \{1, \ldots, \lfloor 2\sqrt{n} + 2 \rfloor\}$*
- *Set the remaining bits to 0*
- *We show that for any shift from 0 to n, two strings of the length W "meet" with probability 1*

Algorithm for many processors

- The main observation is that if there are many processors, each one must use its radio much less!
- We, in fact, show nearly matching (within poly-log) upper and lower bounds.
- Main insight: make every processor connect to only a few other random processors. Since there are lots of processors, this will happen with much fewer radio invocations.

Algorithm for many processors

- Why is connecting to only a few other random processors is a good idea?
- This creates an expander graph with high probability, where adjacent nodes are synchronized.
- We can now run classical synchronization protocols over expander graph.
- Small diameter of expander $\Rightarrow$ fast synchronization!

Algorithm for many processors

- Lower bond for $m > 2$ is more involved, see the paper.
- Our results also extend to the radio broadcast model where in the case of interference only noise is heard.

Protocol that does not need to know m (the number of processors)

- Why should we care about not knowing m?
- If m is large, radio use is also small, so we do not have to spend lots of energy! The bigger m, the less energy (per device) we need to spend communicating.
- How do we estimate m?

Protocol that does not need to know m (the number of processors)

- First, assume that m is large, say $m = n \cdot \text{poly-log}(n)$.
- If we did not underestimate m, we could run the protocol for known m and then *count* the number of nodes.
- If the number of nodes is greater than our estimate of m we are done.
- If not, cut the estimate of m and try again. The key observation is the energy of the next round is far greater then all the previous rounds combined, so the most expensive (last round) is the only one that counts!

Conclusion and Open Questions

- 2-processor case:
  - Matching deterministic upper and lower bound.
- Multi-processor case:
  - Lower bound for a deterministic protocol.
  - Nearly matching upper bound for a randomized protocol.
- Questions:
  - Deterministic protocols for multi-processors case?
  - Close the gap for randomized setting.
  - We can ask many other distributed computing questions in this model.

Thank You