Communication Using Faulty Beeps

Avery Miller
University of Manitoba

Kokouvi Hounkanli
Université du Québec en Outaouais

Andrzej Pelc
Université du Québec en Outaouais

Talk Outline

1. A Message-Passing Model (barely...)
2. Motivation
3. Literature
4. Synchronization and Consensus in a Faulty MAC
5. Open Questions

Beeping

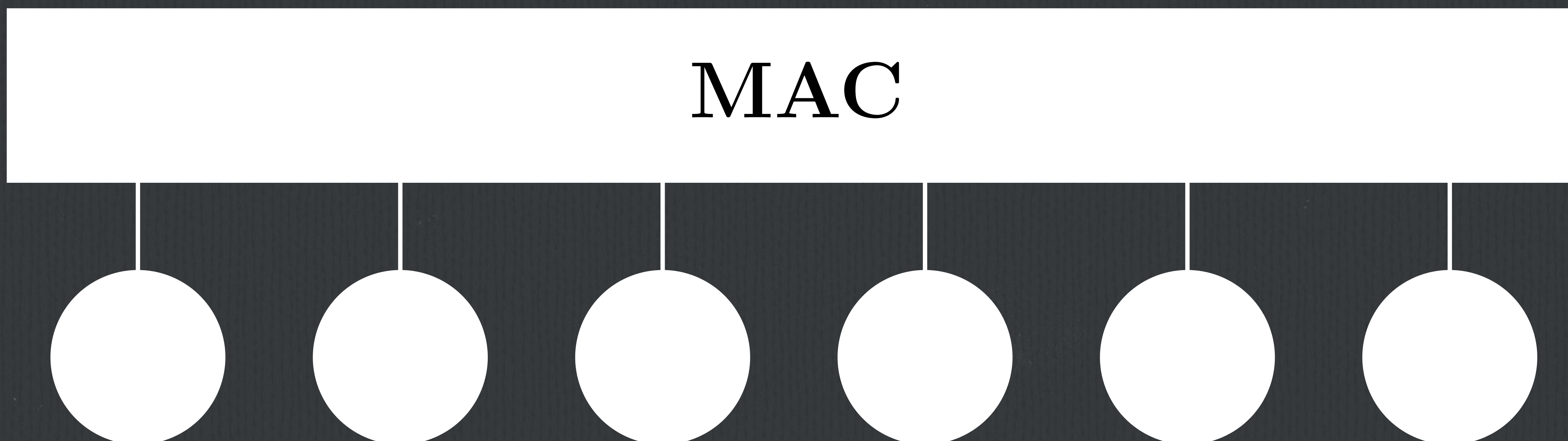
- A network of n processes, modeled as a graph
- Time is synchronous: slots with aligned boundaries, no global clock
- In each slot, a process can transmit or listen
- Rather than transmitting a binary string, each transmission is a beep

Beeping

- In slot t,
  - if process v is beeping, then it receives nothing in slot t
  - if process v is listening, and no neighbouring process beeps, then v receives nothing in slot t
  - if process v is listening, and one or more neighbouring processes beep, then v receives a beep in slot t
- Note: this is less communication than sending single-bit messages

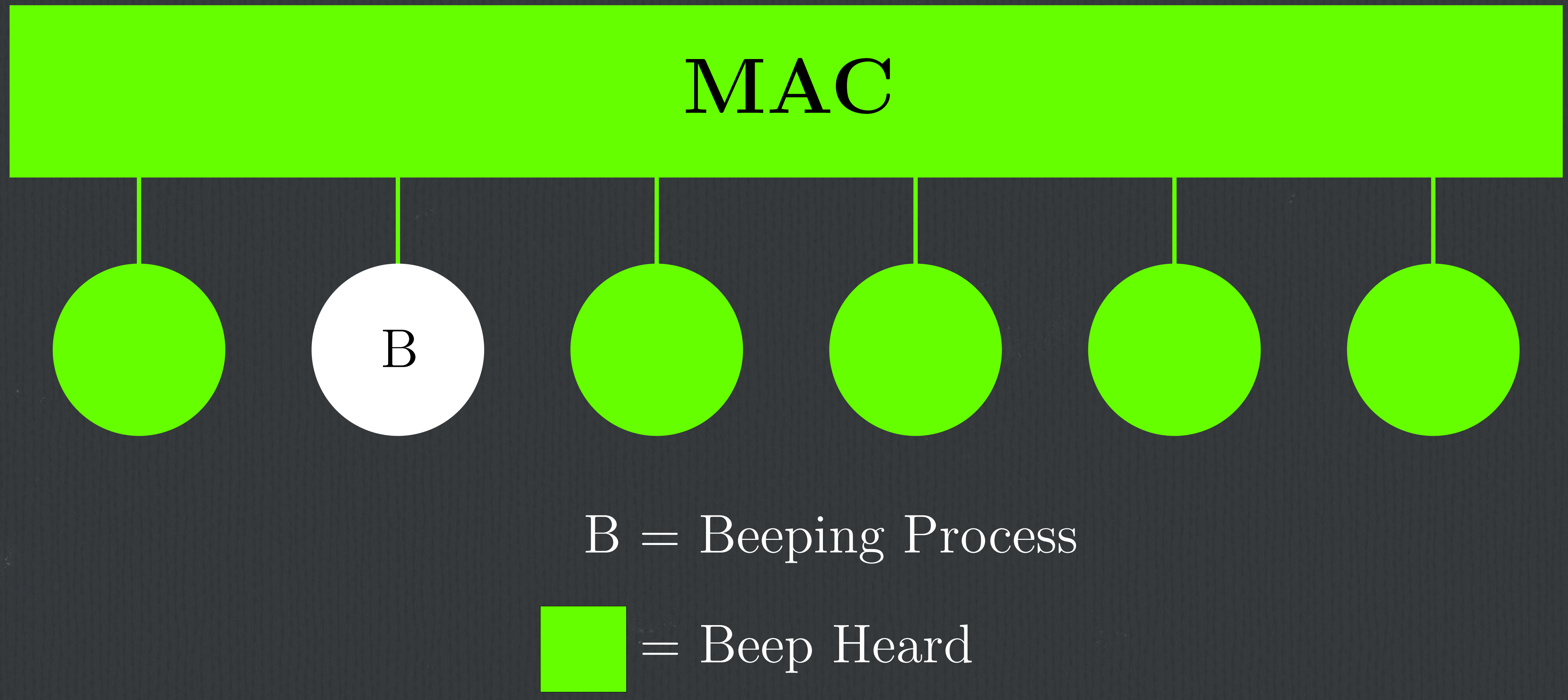
Limitations

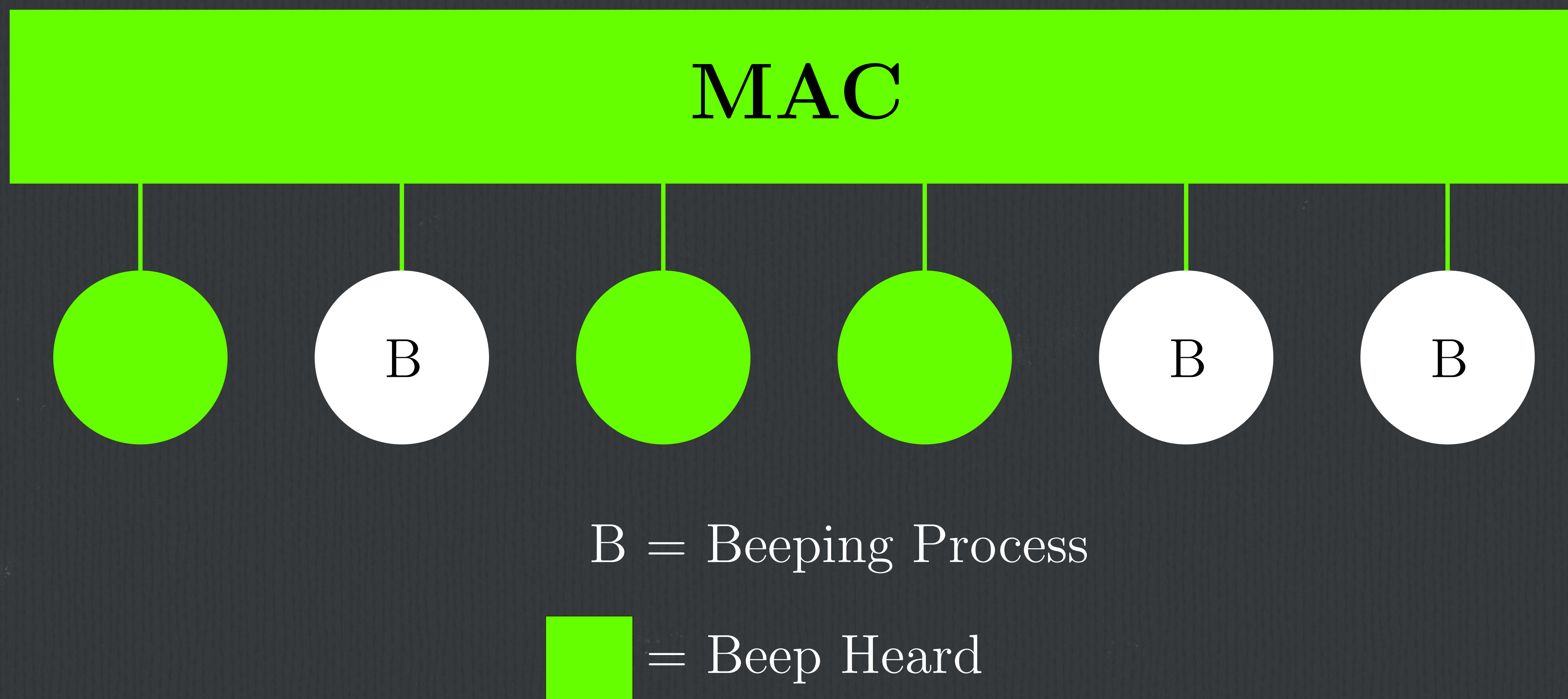
- Beeps are not additive: simultaneous beeps aren't "louder" than a single beep, so can't distinguish number of transmitting neighbours
- Even if only one process beeps, the identity of the transmitter is not known
- The only thing a listening process can distinguish: do I hear noise or not?
- A transmitting process cannot distinguish anything.



B = Beeping Process

= Beep Heard


MAC
B
B = Beeping Process
= Beep Heard


MAC
B
B
B
B = Beeping Process
= Beep Heard

Talk Outline

1. A Message-Passing Model (barely...)
2. <u>Motivation</u>
3. Literature
4. Synchronization and Consensus in a Faulty MAC
5. Open Questions

Why?

- Curiosity: what can we do with the most primitive kind of communication?
- Application to challenging environments: what can we accomplish when very little spectrum/bandwidth/infrastructure is available? Jamming?
- Modeling natural processes: very primitive coordination/communication primitives in nature, which arguably don't send binary strings (fireflies, neurons, others?)

Talk Outline

1. A Message-Passing Model (barely...)
2. Motivation
3. <u>Literature</u>
4. Synchronization and Consensus in a Faulty MAC
5. Open Questions

Results in Beeping Model

- Interval colouring [Cornejo, Kuhn DISC 2010]
- Maximum Independent Set [Afek et al. DISC 2011]
- Conflict Resolution, Membership Problem [Huang, Moscibroda DISC 2013]
- Leader election [Ghaffari, Haeupler SODA 2013] [Förster et al. DISC 2014]
- Broadcast, Multi-broadcast, Gossiping [Czumaj, Davies OPODIS 2015]
- Dominating Set [Yu et al. INFOCOM 2015]
- Randomized space complexity [Gilbert, Newport DISC 2015]

Talk Outline

1. A Message-Passing Model (barely...)
2. Motivation
3. Literature
4. <u>Synchronization and Consensus in a Faulty MAC</u>
5. Open Questions

Global Synchronization

- Consider a multiple-access channel (a clique network) with anonymous processes.
- Our assumption: an adversary may wake up any set of processes in any slot. Upon waking up, each process starts executing its algorithm immediately with local clock equal to 0.
- Our goal: A deterministic algorithm that terminates at all n processes at the same global time. The first slot after termination is time 0 of an established global clock.

Model: Wake-up

- Two (distinguishable) kinds of wake-up

  1. Adversary

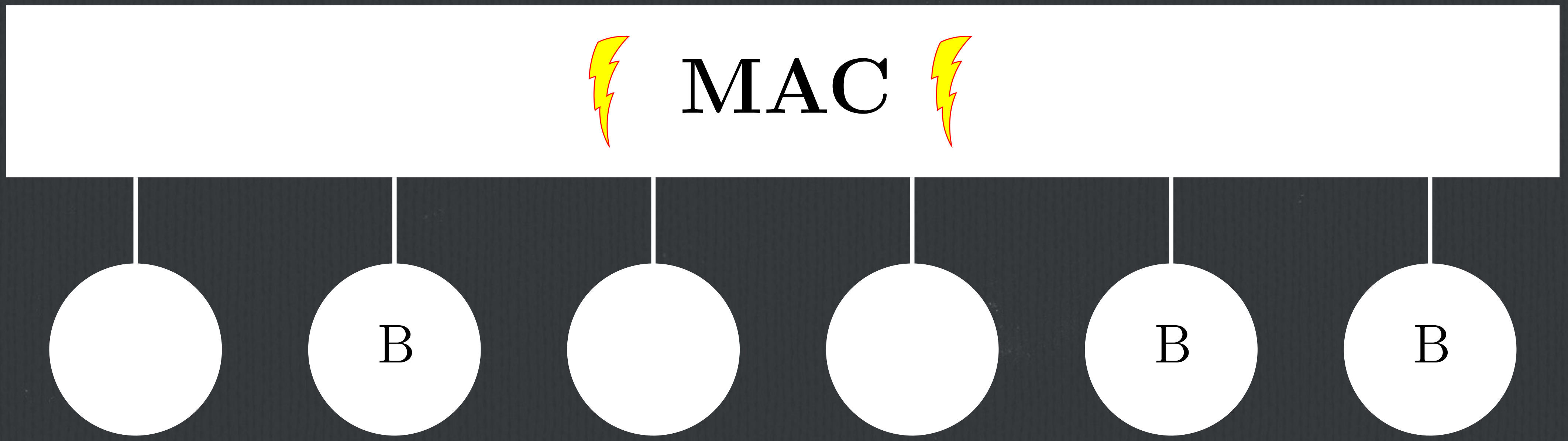
  2. Transmission on channel

- Only adversary wake-up ⇨ impossible

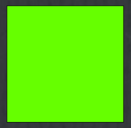
- Add wake-up by transmission ⇨ trivial

Model: Faults

- In each slot, with known constant probability *p*, the channel can <u>fail</u>
- Failure slot:
  - No beeps are heard
  - Transmitter: never knows whether beep succeeded
  - Receiver: can't tell apart silence vs. failure



B = Beeping Process



Challenges

- Initial idea: Synchronize to first heard beep, time t
- Issue: Beeper at time t does not know if successful…
- Solution? Provide feedback at some time t’ > t
- Issue: Beeper at t’ does not know if successful…

… ad infinitum?

- Issue: distinguish between “first” vs. “feedback” beep?

More Challenges

- Processes are anonymous.
  - Issue: What if adversary wakes up all processes at the same time?
- Number of processes is unknown.
  - Issue: What if process is alone?

The GLOBALSYNC Algorithm

- **Theorem:** Fix any constant $\epsilon > 0$. With probability at least $1-\epsilon$, all processes terminate GLOBALSYNC in the same global round, which occurs $O(1)$ rounds after the first wake-up.

- That is, our algorithm runs in constant time and fails with probability at most ϵ for any given constant $\epsilon > 0$.

(multiply all bounds in this talk by (log ε)-factor if you want the bound in terms of ε)

The GLOBALSYNC Algorithm

- Alarm Beeps: when woken up spontaneously, beep periodically, trying to wake up other processes.
- The interval between consecutive alarm beeps increases, to avoid clever adversary.
- Between alarm beeps, wait and listen for feedback beeps.
- If large number of unanswered alarm beeps, terminate algorithm at next scheduled alarm beep.
  W.h.p: lone process or all woken up at same time

The GLOBALSYNC Algorithm

- Let constant $\gamma > 0$ such that $p^{\gamma} < \epsilon/4$.
- Remark: in a sequence of γ consecutive beeps, at least one occurs in a fault-free round with high probability.
- When alarm beeper hears feedback:
  listen 2γ rounds then beeps for 2γ rounds.
- When woken up by a beep:
  listen 2γ rounds and then beep for 2γ rounds.

The GLOBALSYNC Algorithm

- Terminate in round $r+4\gamma+1$, where r is first successful alarm beep.

<u>What is r?</u>

- Divide time into phases of length 2γ. Let t be first beep heard by v.
  - Single beep in the phase $\Rightarrow$ t is alarm beep
  - Multiple beeps in the phase $\Rightarrow$ t is feedback beep
- If t is alarm beep, v sets $r = t$.
  Else, v sets r to be its most recent transmit round.

Application: (Weak) Consensus

- The task:
  - Each process receives an input value
  - All processes must output the same value
  - Validity: if all inputs are the same, then output = input
  - In general, output can differ from all inputs (no integrity)

The Algorithm

- Essentially set disjointness: as soon as the processes detect two different inputs, they output a known default value. If no difference detected, all processes output their own input value.
- First, synchronize clocks using `GLOBALSYNC`
- Each process beeps out its own input value in binary: 1 = beep, 0 = listen
- As we did in `GLOBALSYNC`, proceed in rounds of length γ to ensure that a successful beep occurs in each round with high probability.
- As soon difference is spotted, output default value, else, output own input value

Encoding

- For this to work, need to re-encode input values
- replace each 0 with 01, each 1 with 10, end with 11
- It follows that:
  - No encoded value is a prefix of another
  - At the first bit where original inputs differ, all nodes hear a beep in one of the two corresponding rounds

Bounds

- Theorem: Fix any constant $\epsilon > 0$. With error probability at most ϵ, consensus can be solved deterministically using beeps in a fault-prone MAC in time $O(\log w)$, where w is the smallest of all input values of processes in the channel.
- Theorem: Deterministic consensus in a fault-free MAC with beeps requires $\Omega(\log w)$ rounds, where w is the smallest of all input values of processes in the channel.

Talk Outline

1. A Message-Passing Model (barely...)
2. Motivation
3. Literature
4. Synchronization and Consensus in a Faulty MAC
5. <u>Open Questions</u>

Open Questions

- Other fault models:
  - individual failures: a beep is not sent or not received
  - failure probability p is not known
  - adversarial
- Multi-hop network instead of MAC
- Randomized algorithms
- Consensus with strong validity