Formal verification of a synchronization algorithm

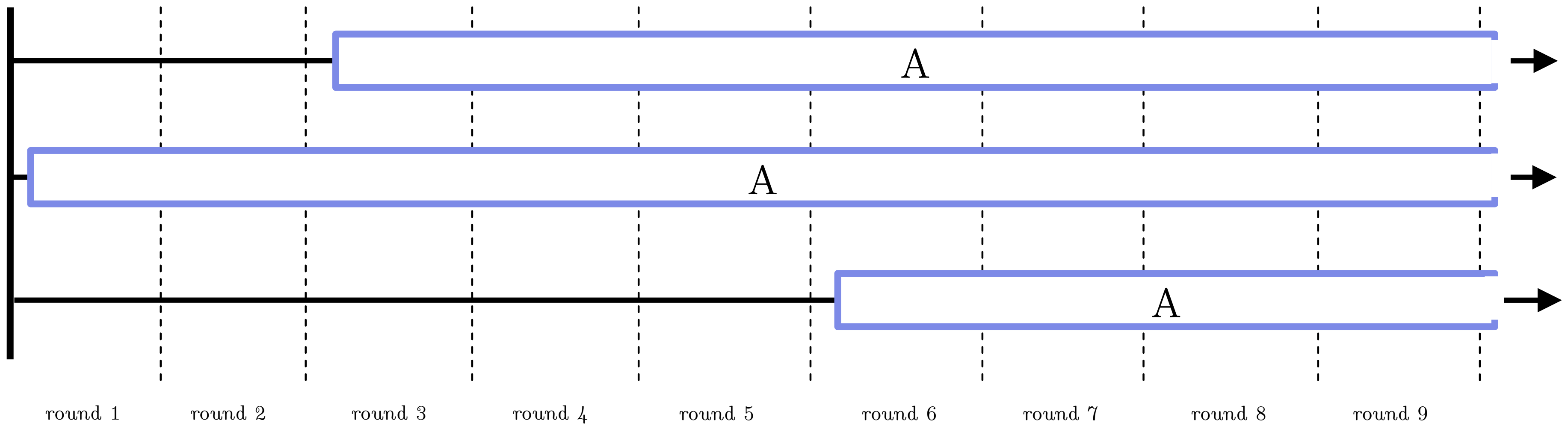
Louis Penet de Monterno

LIX, École polytechnique, IP Paris

The model

- Nodes operate in synchronous rounds
- Asynchronous starts: nodes do not start in the same round

The model

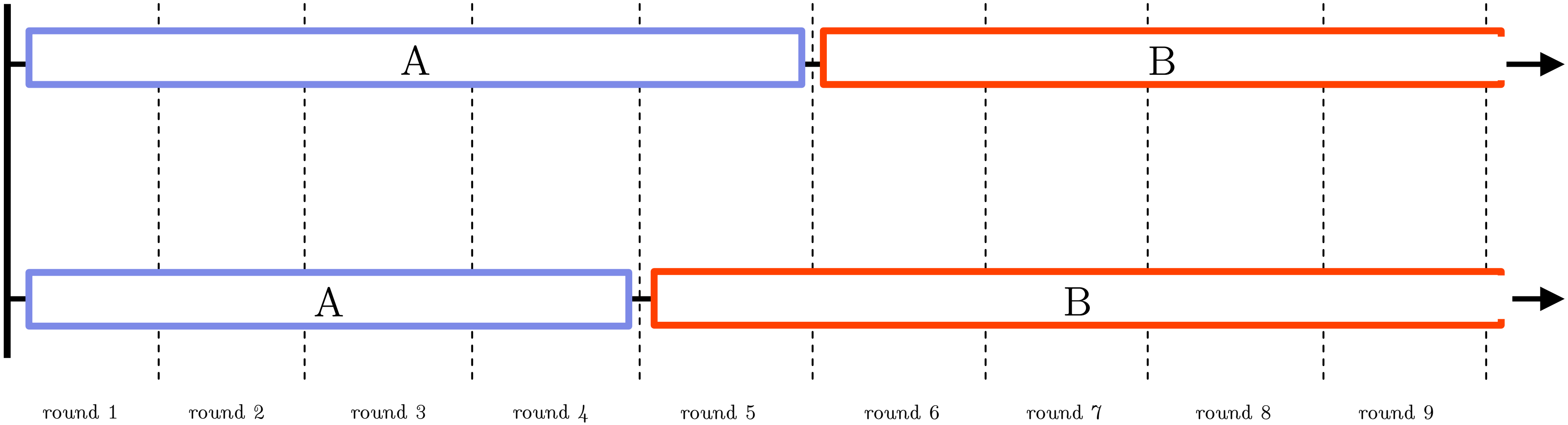

A
A
A
round 1
round 2
round 3
round 4
round 5
round 6
round 7
round 8
round 9

Justifying the problem

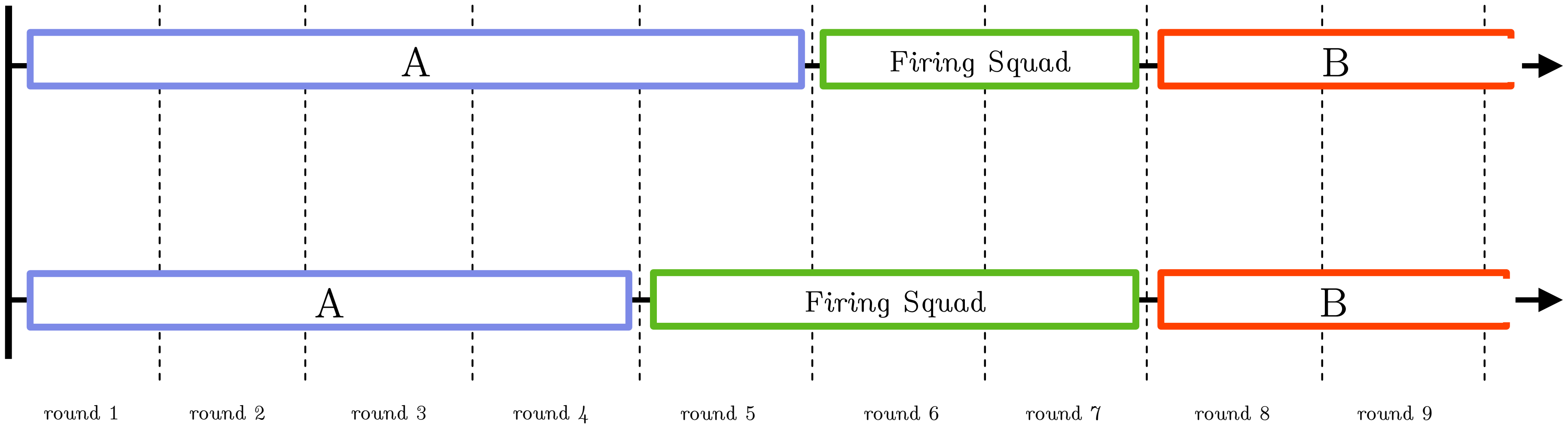
- Execution of a sequence of algorithms $A\,;B$

Justifying the problem

- Execution of a sequence of algorithms $A; B$



A first solution: the firing squad algorithm



The firing squad problem

- Liveness: every node eventually "fires"
- Safety: if two nodes "fire", they "fires" simultaneously (in the same round)

The firing squad problem

- Liveness: every node eventually "fires"
- Safety: if two nodes "fire", they "fires" simultaneously (in the same round)

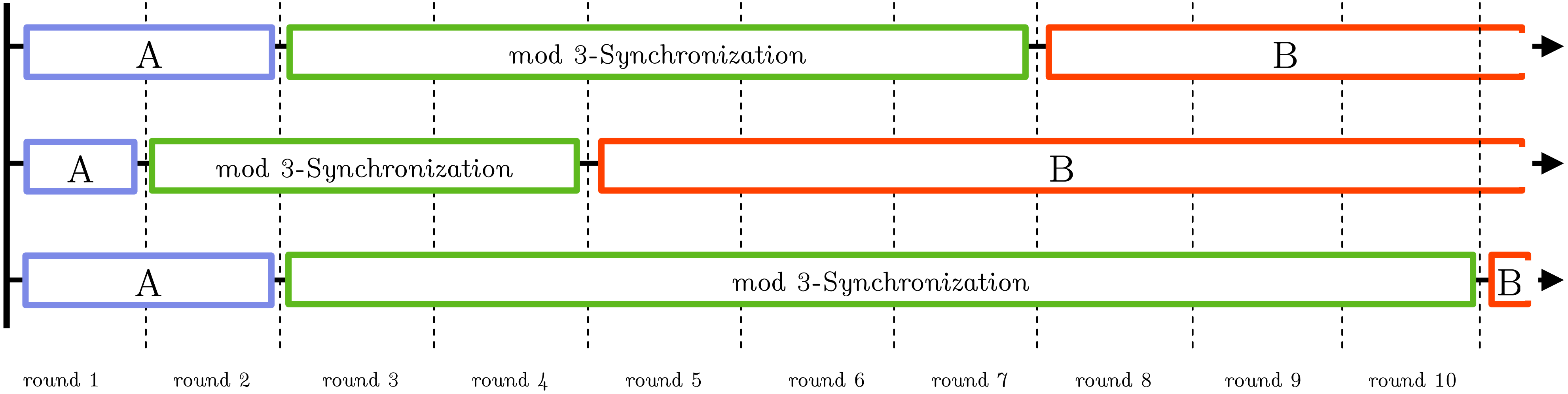
→ Essentially unsolvable without strong connectivity in each round

[Charron-Bost & Moran. TCS2019]

The mod P-synchronization problem

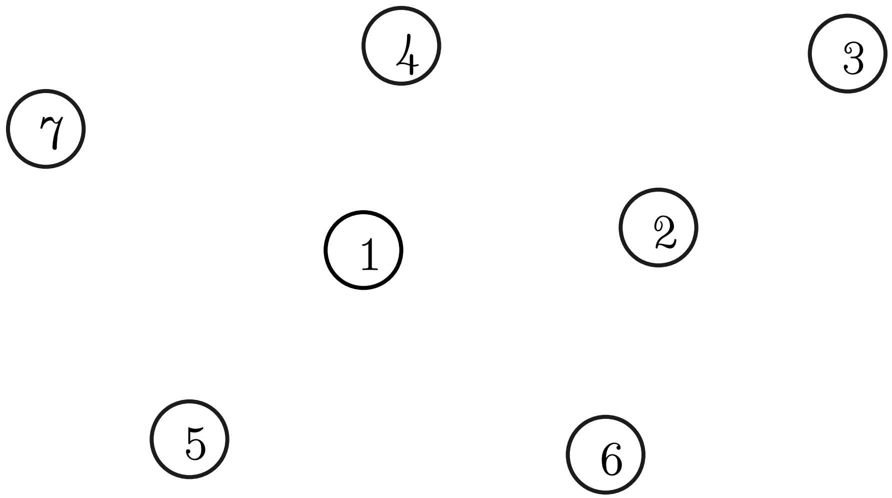
- Liveness: each node eventually "fires"
- Safety: if two node "fire", they "fire" in the same round modulo P

The mod P-synchronization problem

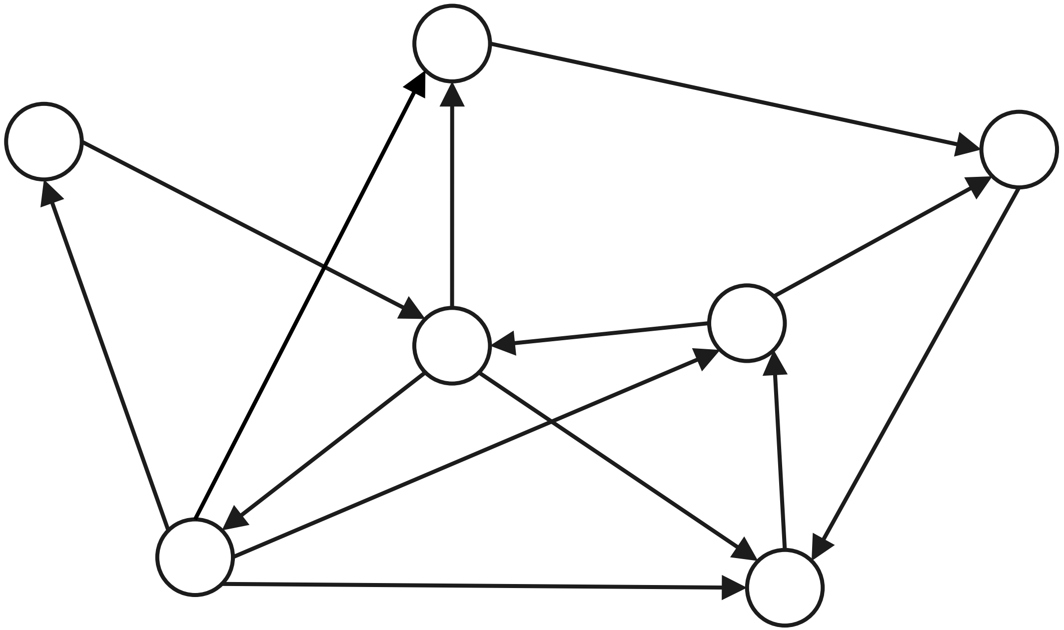


Uses cases of mod P-firing squad

- Round-robin leader election
  - Round 1: node 1 leads
  - Round 2: node 2 leads
  - ...
  - Round 7: node 7 leads
  - Round 8: node 1 leads again
  - ...
- $P = n$

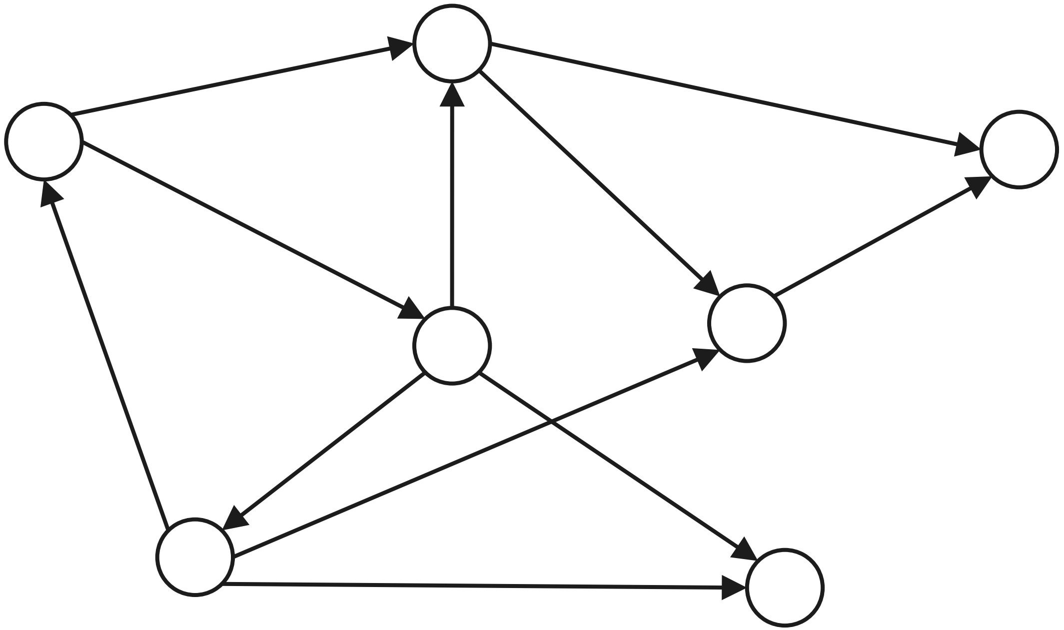


The model



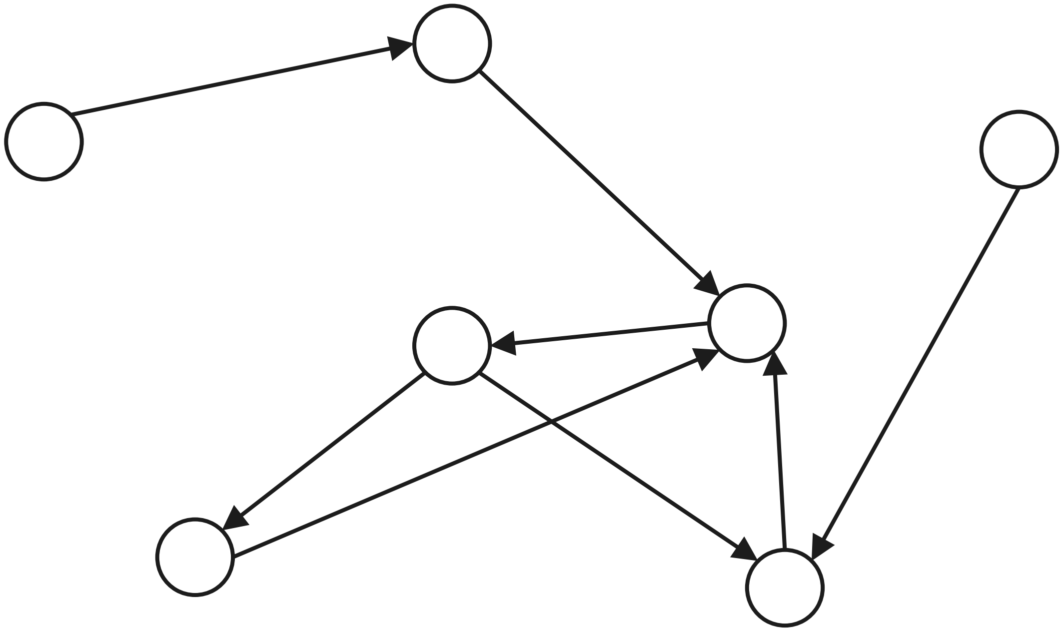
round 1

The model



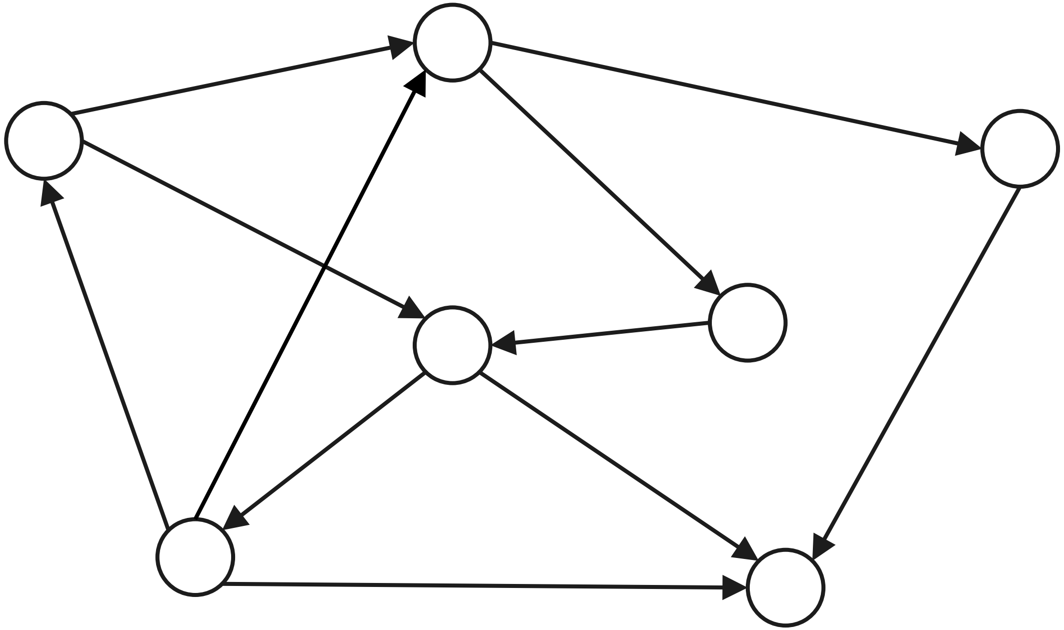
round 2

The model



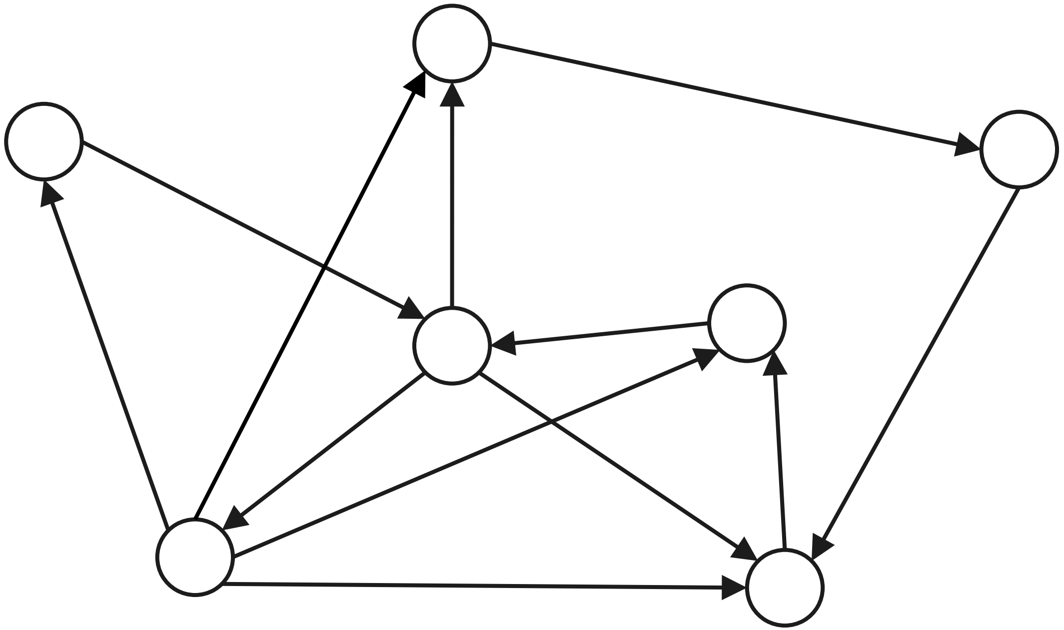
round 3

The model



round 4

The model

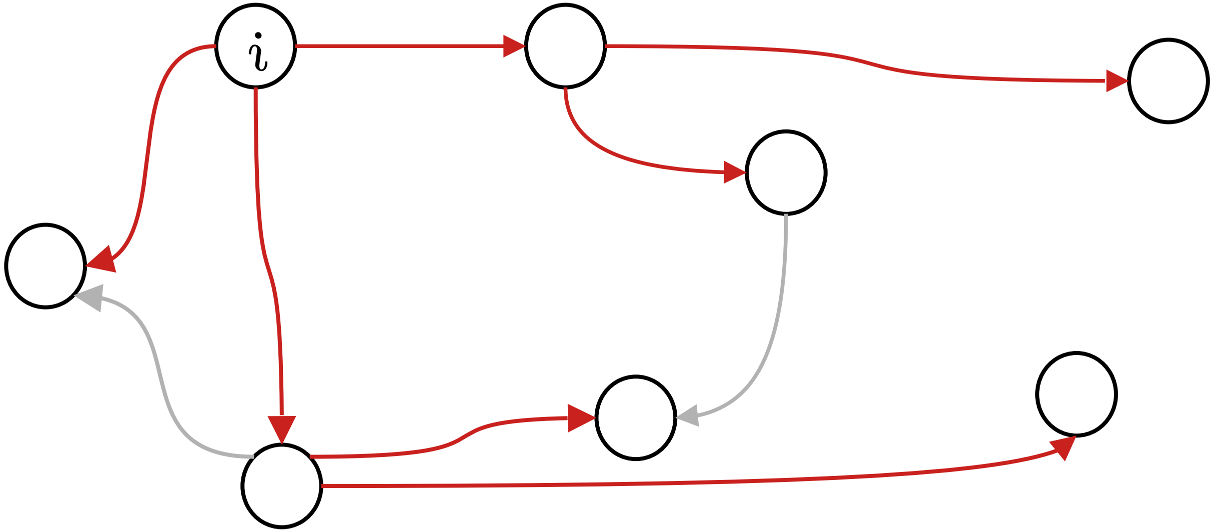


round 5

Our result

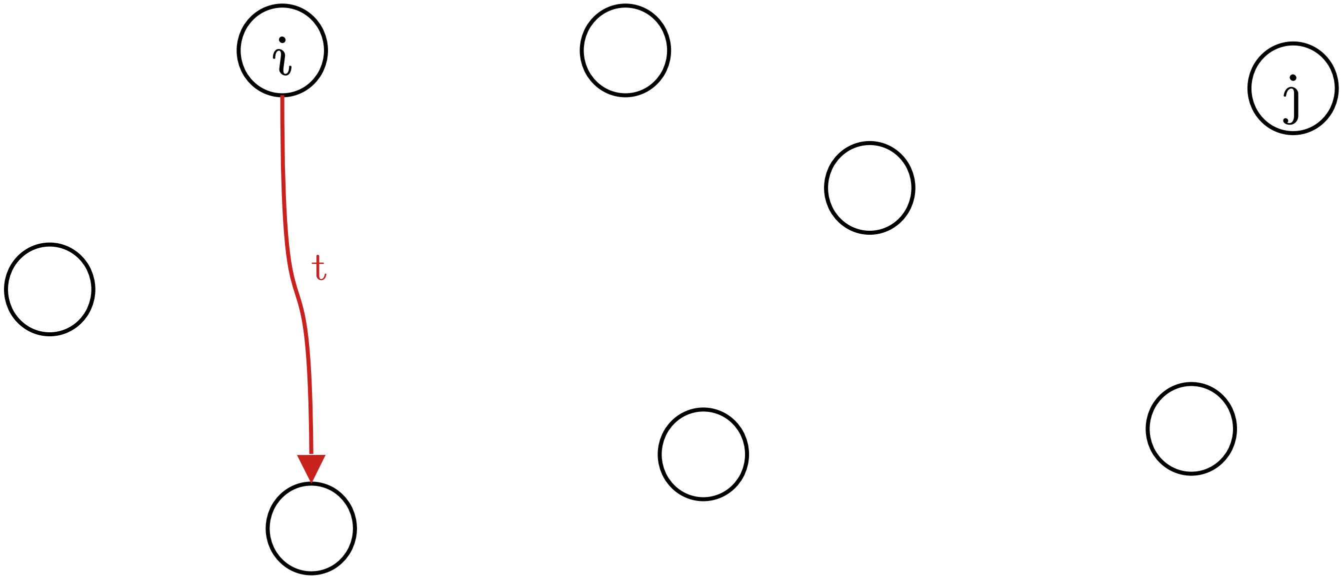
- We solve mod P-synchronization assuming that:
  - The dynamic radius, denoted R, is finite.
  - The nodes must "know" an upper bound on R.

Eccentricity of node i



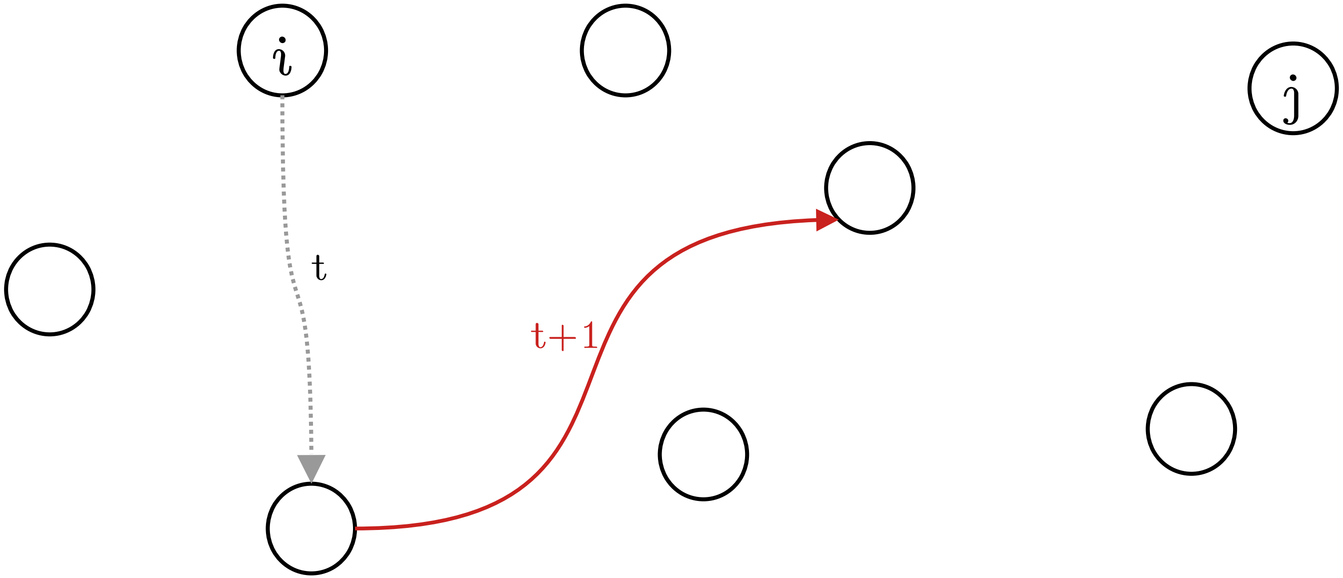
- The eccentricity of i is 2
- All other eccentricities are infinite
- i is said to be central
- The radius is 2

A temporal path in a dynamic graph



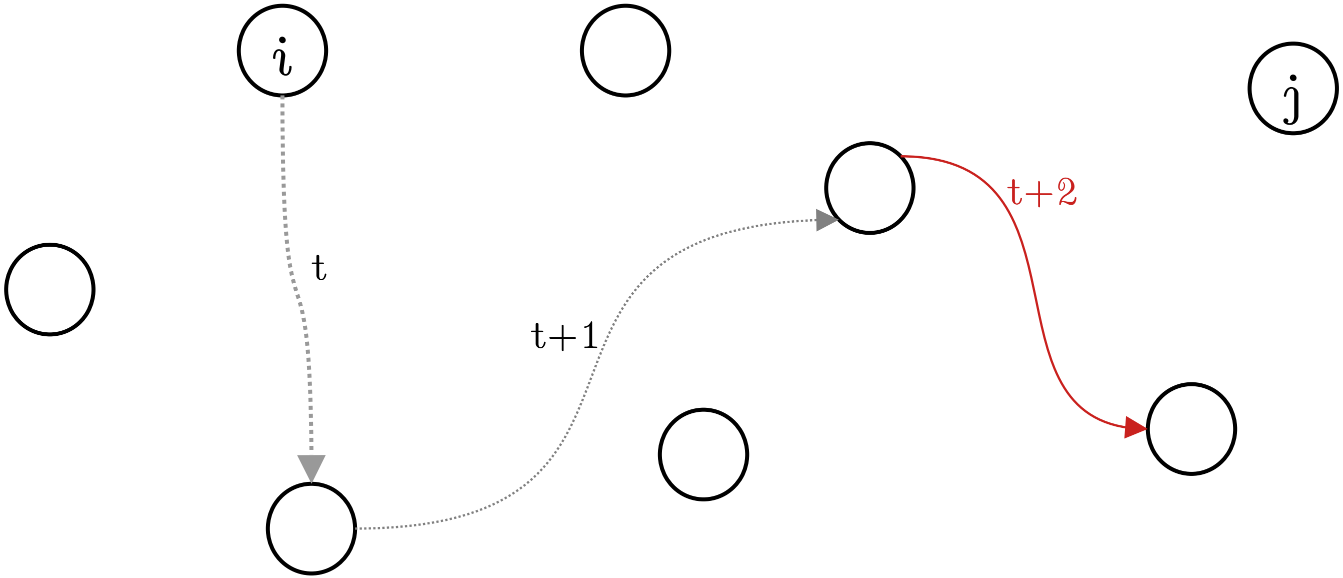
Round t

A temporal path in a dynamic graph



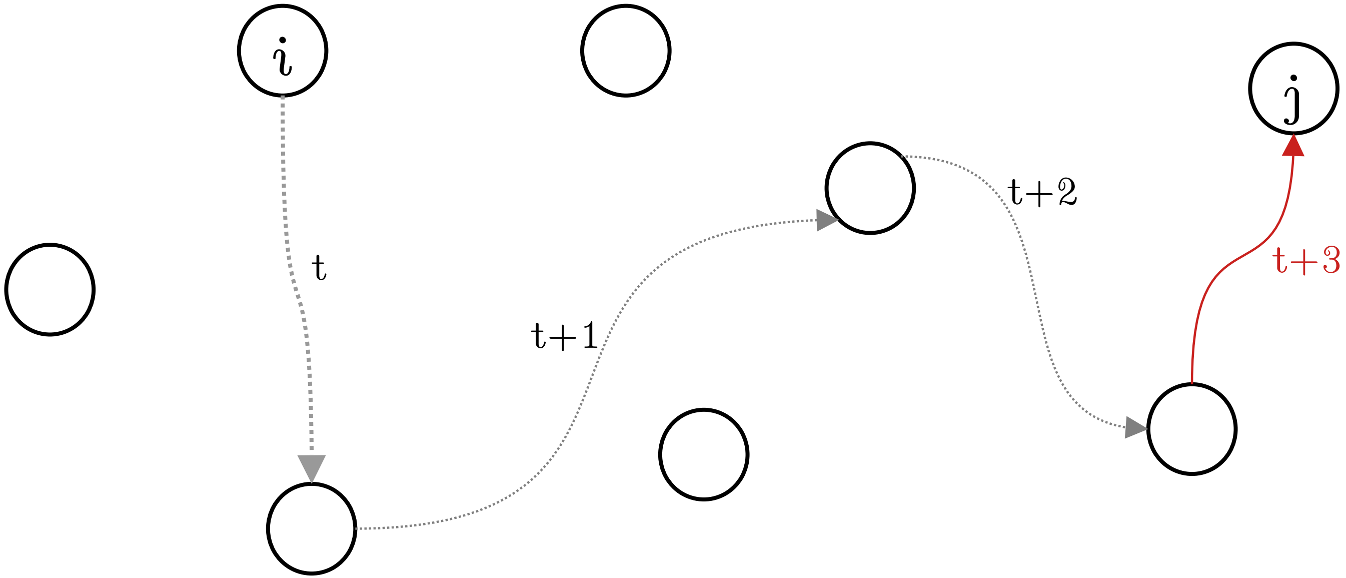
Round t+1

A temporal path in a dynamic graph



Round t+2

A temporal path in a dynamic graph



Round t+3

Dynamic eccentricity and dynamic radius

- The eccentricity of node i is the longest shortest path between i and any other node.
- The dynamic eccentricity of node i is the longest shortest temporal path between i and any other node.

- The radius of a graph is the minimum eccentricity.
- The dynamic radius of a graph is the minimum dynamic eccentricity.

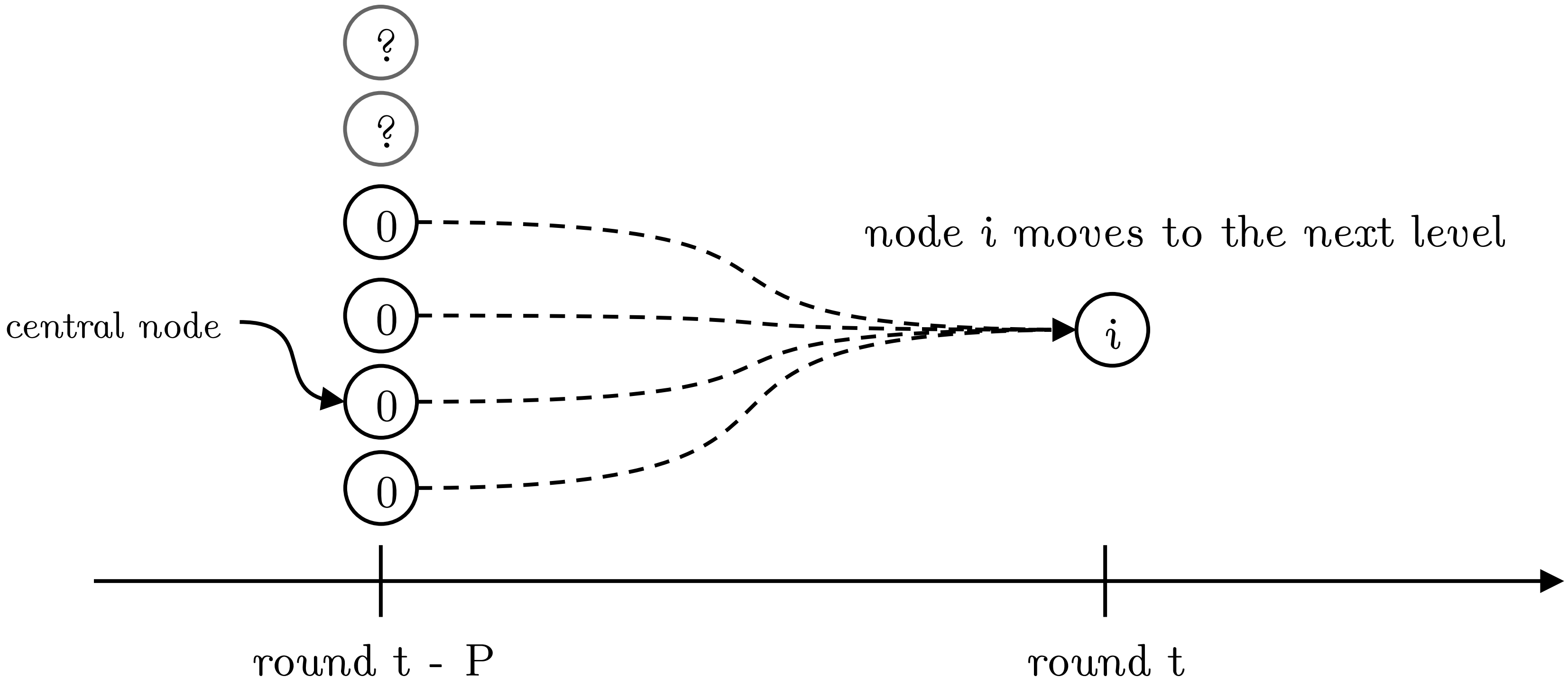
Intermediary result

- We construct the algorithm $SynchMod_P$ which solves mod P-synchronization, assuming a finite dynamic radius R and
  - $R \leq P$
  - $P > 2$

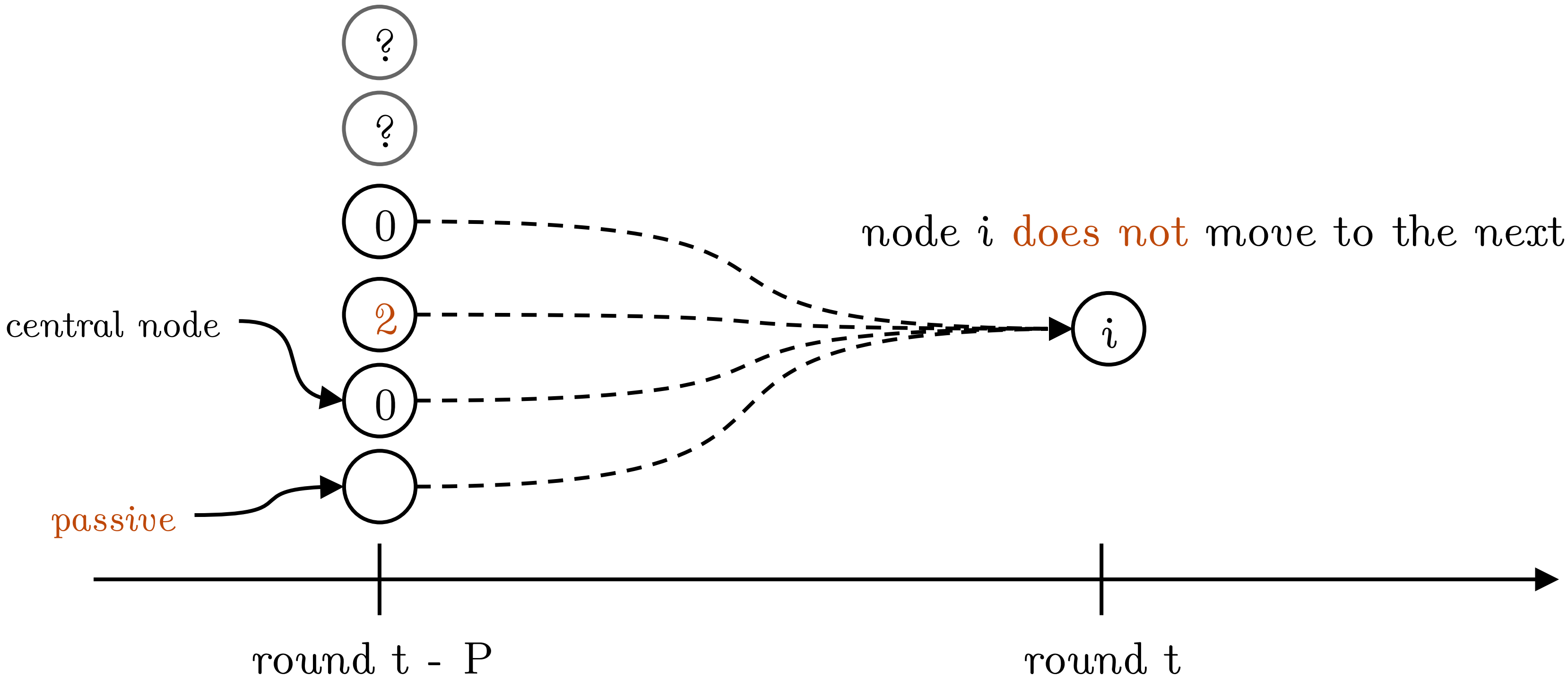
Presentation of $\mathrm{SynchMod}_{\mathrm{P}}$

- Each node i holds a variable $level_i \in \{0,1,2\}$
  - Initial level = level 0
  - Reaching level 2 = firing
- Each node i holds a variable $c_i \in \mathbb{N}$

Presentation of SynchMod_P



Presentation of SynchMod_P



Presentation of SynchMod_P

- Initially:
  - $c_i = 0$
  - $level_i = 0$
  - $force_i = 0$
  - $synch_i = false$
  - $ready_i = false$
- At each round:
  - send $<c_i, force_i, synch_i, ready_i>$ to all
  - receive messages from nodes In^a
  - if all receives messages are non-null
    - $synch_i \leftarrow \bigwedge synch_j \wedge c_j \equiv_P c_i$
  - else
    - $synch_i \leftarrow false$
  - $ready_i \leftarrow \bigwedge ready_j$
  - $force_i \leftarrow \max force_j$
  - $c_i \leftarrow 1 + \min c_j$
  - if $c_i \equiv_P 0$
    - If $synch_i \wedge level_i = 0$
      - $level_i \leftarrow 1$
      - if $force_i < 2$
        - $force_i \leftarrow 1$
        - $c_i \leftarrow 0$
    - if $level_i = 1 \wedge ready_i \wedge synch_i$
      - $force_i \leftarrow 2$
      - $level_i \leftarrow 2$
      - $c_i \leftarrow 0$
    - $synch_i \leftarrow true$
    - $ready_i \leftarrow level > 0$

Getting rid of extra assumptions

Solving mod P-synchronization

⇓

Solving mod P'-synchronization

where P' is a divisor of P

Example: if the dynamic radius is at most equal to 7 and P = 3, then:

- SynchMod_3 is incorrect
- SynchMod_9 solves mod 9-synchronization, and thus mod 3-synchronization

An impossibility result

Theorem:

> The mod P-synchronization problem is unsolvable if the dynamic radius is finite but no bound on it is known by the nodes

Complexity of SynchMod$_P$

- Time-Complexity: $O(n)$
- Memory storage of each node:
  - Initial approach: memory usage tends to infinity
  - Optimized approach: $O(\log n)$

n is the size of the network

the parameter P is treated as a constant

The model

- `datatype 's state =`
  `| Active s`
  `| Passive`
- `exec :: nat ⇒ Node ⇒ s state`
- `network :: nat ⇒ Node ⇒ Node set`

set of states
of the algorithm

The model

- Definition of the algorithm:
  - `InitState :: s ⇒ bool`
  - `SendMsg   :: s ⇒ m`
  - `NextState :: s ⇒ (Node ⇒ m message) ⇒ s ⇒ bool`

- datatype 'm message =

```
          | Content m
          | Void
          | Bot
```

a message received with payload in m

absence of message

“heartbeats” sent by passive node

Formal proof

```
fixes P network exec central_node
constrains P :: nat
  network :: nat ⇒ Proc ⇒ Node Set
  exec :: nat ⇒ Proc ⇒ s state
  central_node :: Proc
assumes star: ∀ i n. path network central_node i n P
  and loop: ∀ i r. i : network r i
  and run: HORun Algo exec network
  and P2: P > 2
  and complete: ∀ i. ∃ t. rho t i ≠ Asleep
  and finite: OFCLASS(proc, finite_class)
```

Formal proof

- theorem /* liveness */
  - ∀ p. ∃ s. rho b p = Active s ∧ level s = 2

- theorem / * safety */
  - ∃ c. ∀ i t s1 s2.

```
rho t i = Active s ⟶
(level s < 2) ⟶
rho (Suc t) i = Active s2 ⟶
level s2 = 2 ⟶
t mod P = c
```

Conclusion

- Our contribution:
  - Definition of the mod P-synchronization problem
  - Introduction of SynchMod_P
  - Verification of SynchMod_P using Isabelle