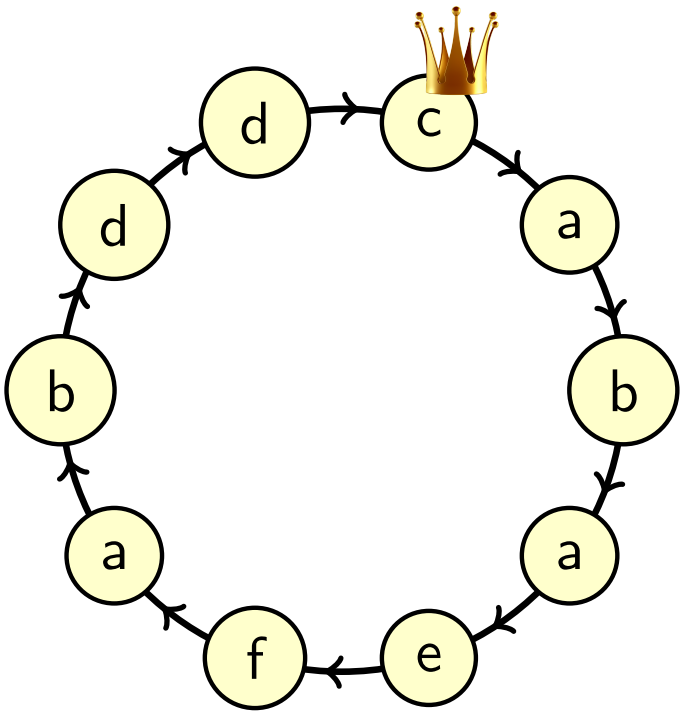
Leader Election in Asymmetric Labeled Unidirectional Rings

Karine Altisen[1] Ajoy K. Datta[2] Stéphane Devismes[1]
Anaïs Durand[1] Lawrence L. Larmore[2]

[1] Univ. Grenoble Alpes, CNRS, Grenoble INP, VERIMAG, 38000 Grenoble, France
[2] University of Nevada Las Vegas, USA

Meeting DESCARTES, October 2-4 2017, Poitiers

Context



- Leader election
- Unidirectional rings
- Homonym processes
- Deterministic algorithm
- Asynchronous message-passing

State of the Art - Leader Election

Anonymous processes

Identified processes

Deterministic solution

Probabilistic solution

Impossible
[Angluin, 80]
[Lynch, 96]

Possible
[Afek and Matias, 94]
[Kutten *et al.*, 13]

Possible
[LeLann, 77]
[Chang and Roberts, 79]
[Peterson, 82]

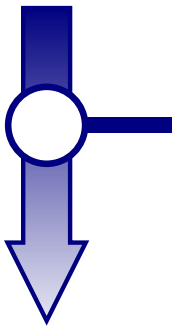
State of the Art - Leader Election

Deterministic solution → **Probabilistic** solution

Anonymous processes

Impossible
[Angluin, 80]
[Lynch, 96]

Possible
[Afek and Matias, 94]
[Kutten *et al.*, 13]



Homonym processes [Yamashita and Kameda, 89]

Identified processes

Possible
[LeLann, 77]
[Chang and Roberts, 79]
[Peterson, 82]

Two versions of the Leader Election problem

1. **Message-terminating:** Processes do not explicitly terminate but only a finite number of messages are exchanged.
2. **Process-terminating:** Every process eventually halts.

Ring Classes

An algorithm A solves the leader election *for the class of ring network* $\mathcal{R}$ if A solves the leader election for every network $R \in \mathcal{R}$.

Ring Classes

An algorithm A solves the leader election *for the class of ring network* $\mathcal{R}$ if A solves the leader election for every network $R \in \mathcal{R}$.

A cannot be given any specific information about the network unless that information holds for all members of $\mathcal{R}$.

Ring Classes

An algorithm A solves the leader election *for the class of ring network* $\mathcal{R}$ if A solves the leader election for every network $R \in \mathcal{R}$.

A cannot be given any specific information about the network unless that information holds for all members of $\mathcal{R}$.

We consider three important classes of ring networks.

- $\mathcal{K}_k$ is the class of all ring networks such that no label occurs **more than** k times.
- $\mathcal{A}$ is the class of all asymmetric ring networks: rings with no non-trivial rotational symmetry.
- $\mathcal{U}^*$ is the class of all rings in which at least one label is unique.

Symmetric *vs.* Asymmetric

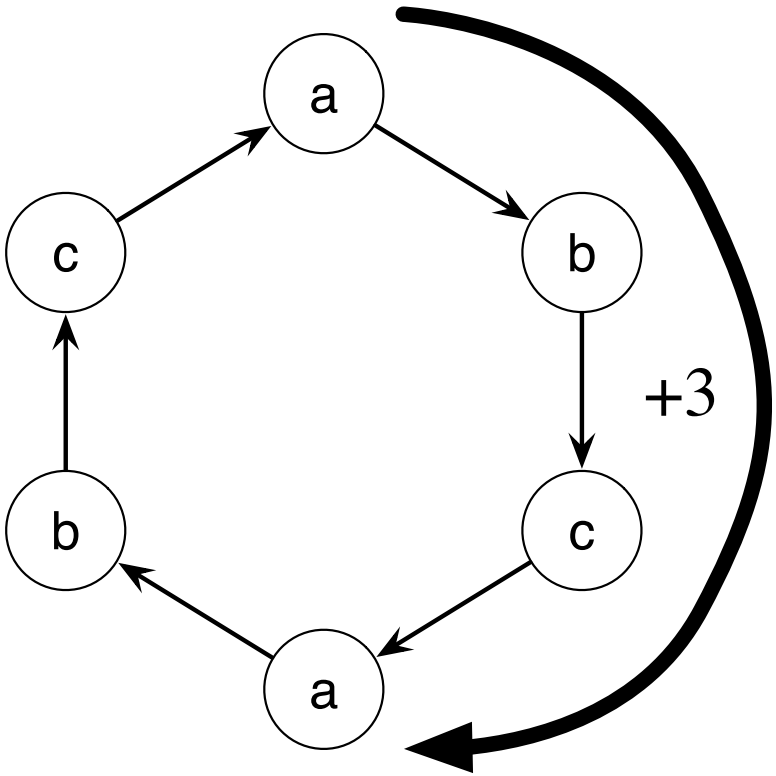


Figure : Symmetric Ring

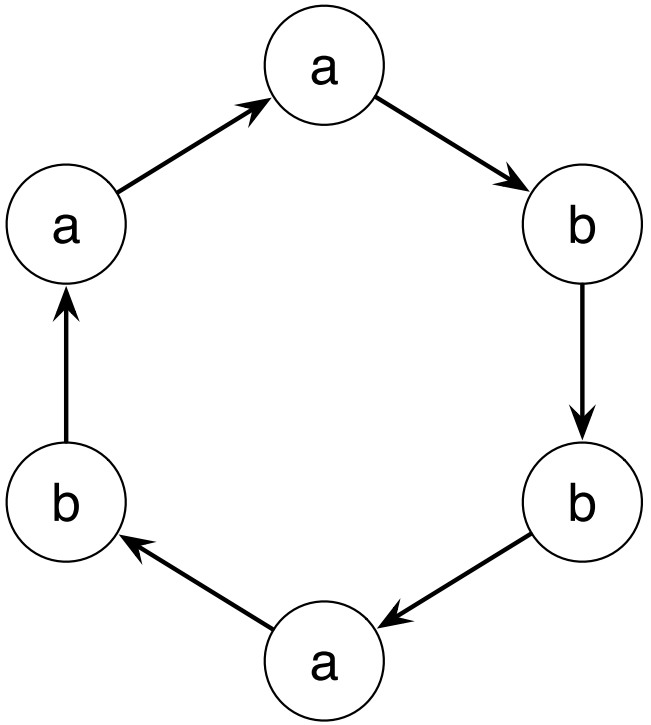


Figure : Asymmetric Ring

Inclusions

- $\mathcal{K}_1 \subset \mathcal{K}_2 \subset \mathcal{K}_3 \ldots$
- $U^* \cap \mathcal{K}_1 \subset U^* \cap \mathcal{K}_2 \subset U^* \cap \mathcal{K}_3 \ldots \subset \mathcal{U}^*$
- $\mathcal{K}_1 \subset \mathcal{U}^*$
- $\mathcal{U}^* \subset \mathcal{A}$

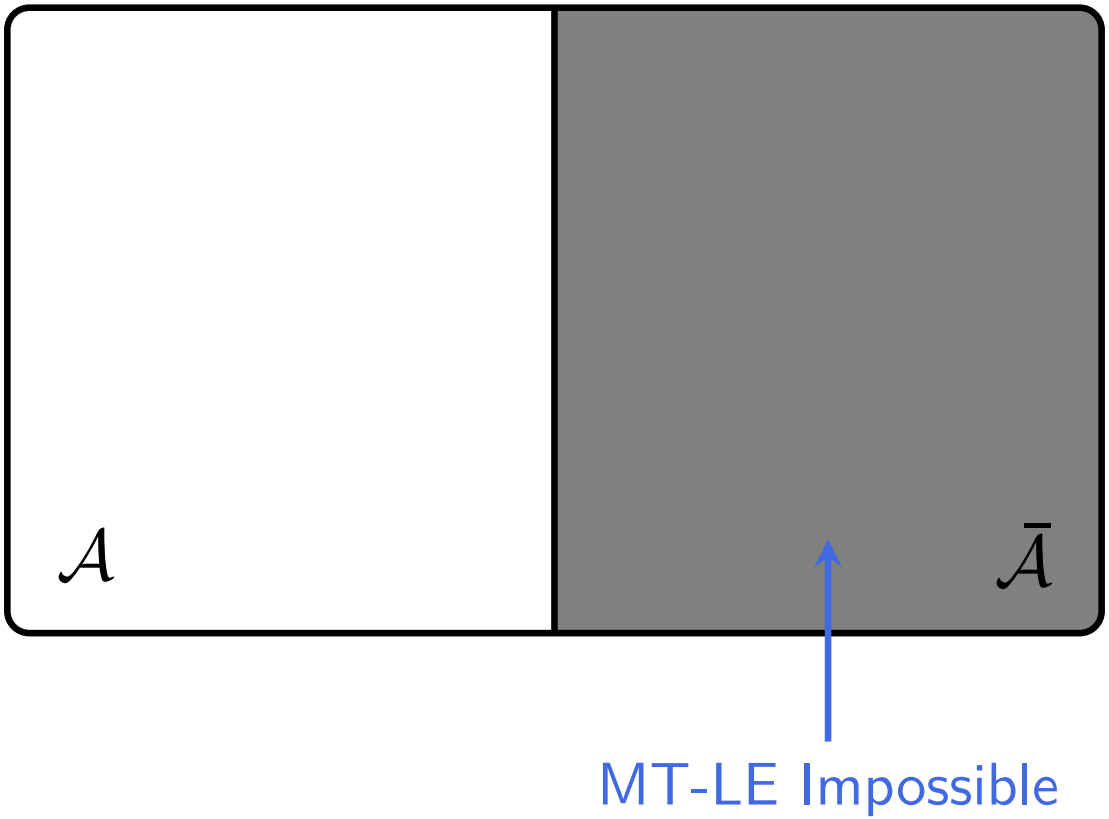
State of the Art *vs.* Contribution

Leader Election in Rings of Homonym Processes

	PT/MT	Asynch.	Uni./Bi.	Known	Ring Class	# Msg	Time
[Delporte *et al.*, 14]	MT	☑	Bi.		# labels > greatest proper divisor of n	?	?
	PT	☑		n		$O(n \log n)$	?
[Dobrev, Pelc, 04]	PT	☒	Bi. + Uni.	$m \leq n$	Decide if inputs are unambiguous	$O(n \log n)$	$O(M)$
		☑	Bi.	$M \geq n$		$O(nM)$	?
[SSS 2016]	PT	☑	Uni.	k	∃ unique label and # proc with same label ≤ k	$O(kn)$	$O(kn)$
[IPDPS 2017]	PT	☑	Uni.	k	Asymmetric labelling and # proc with same label ≤ k	$O(n^2 + kn)$ $O(k^2n^2)$	$O(kn)$ $O(k^2n^2)$

- Uni : Unidirectional / Bi : Bidirectional
- MT = Message-terminating
- PT = Process-terminating

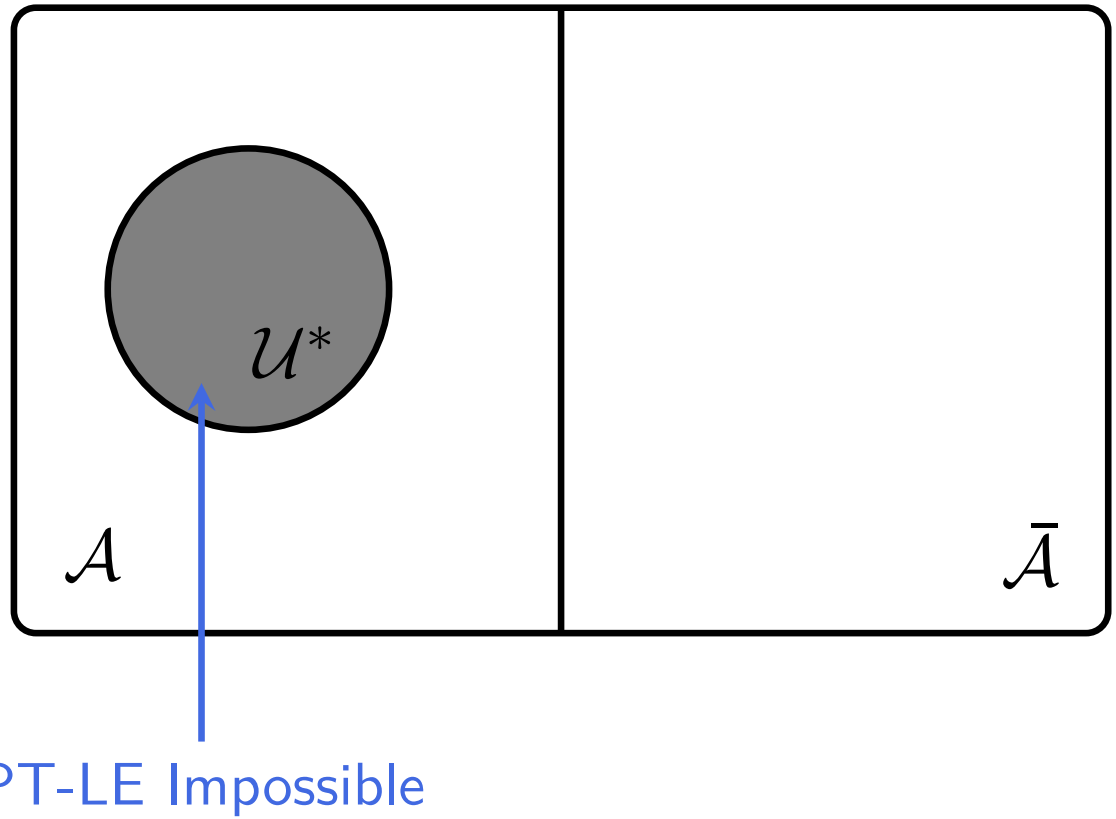
Contributions



- MT-LE: Message-Terminating Leader Election
- PT-LE: Process-Terminating Leader Election

- $\mathcal{A}$: Rings with asymmetric labelling
- $\overline{\mathcal{A}}$: Rings with symmetric labelling
- $\mathcal{U}^*$: Rings with at least one unique label
- $\mathcal{K}_k$: Rings with no more than k processes with the same label

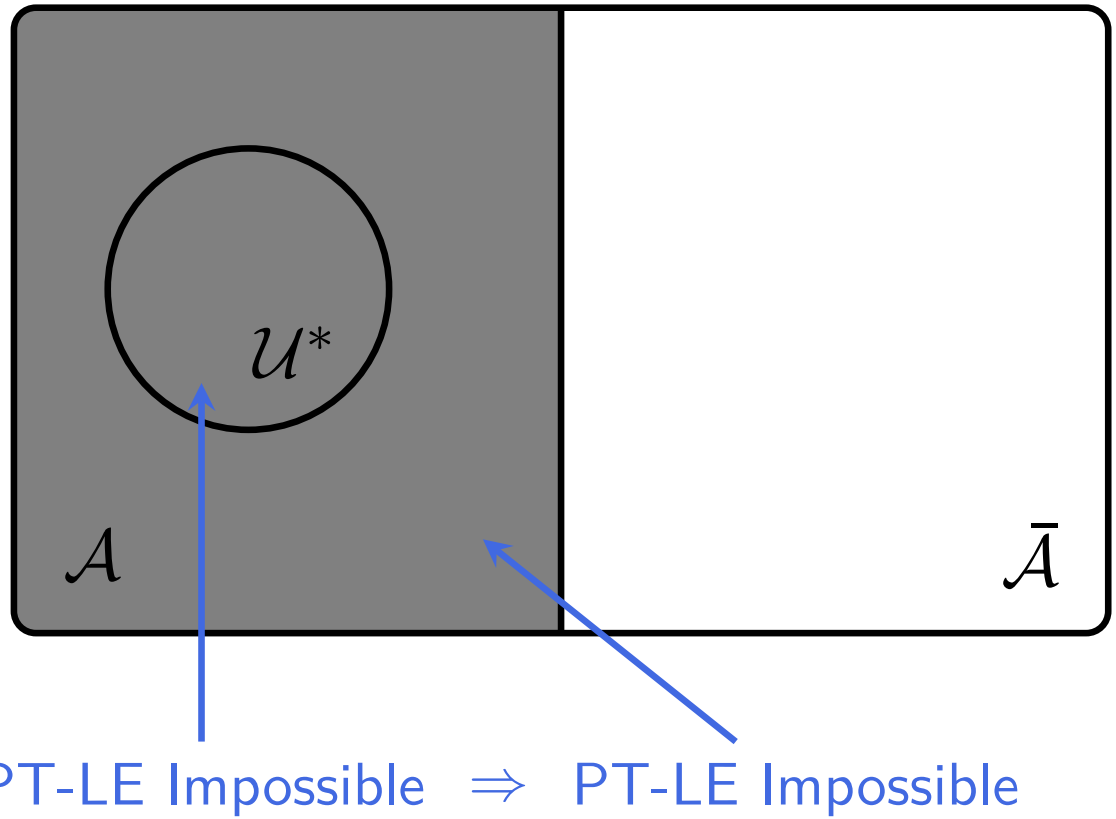
Contributions



- MT-LE: Message-Terminating Leader Election
- PT-LE: Process-Terminating Leader Election

- $\mathcal{A}$: Rings with asymmetric labelling
- $\overline{\mathcal{A}}$: Rings with symmetric labelling
- $\mathcal{U}^*$: Rings with at least one unique label
- $\mathcal{K}_k$: Rings with no more than k processes with the same label

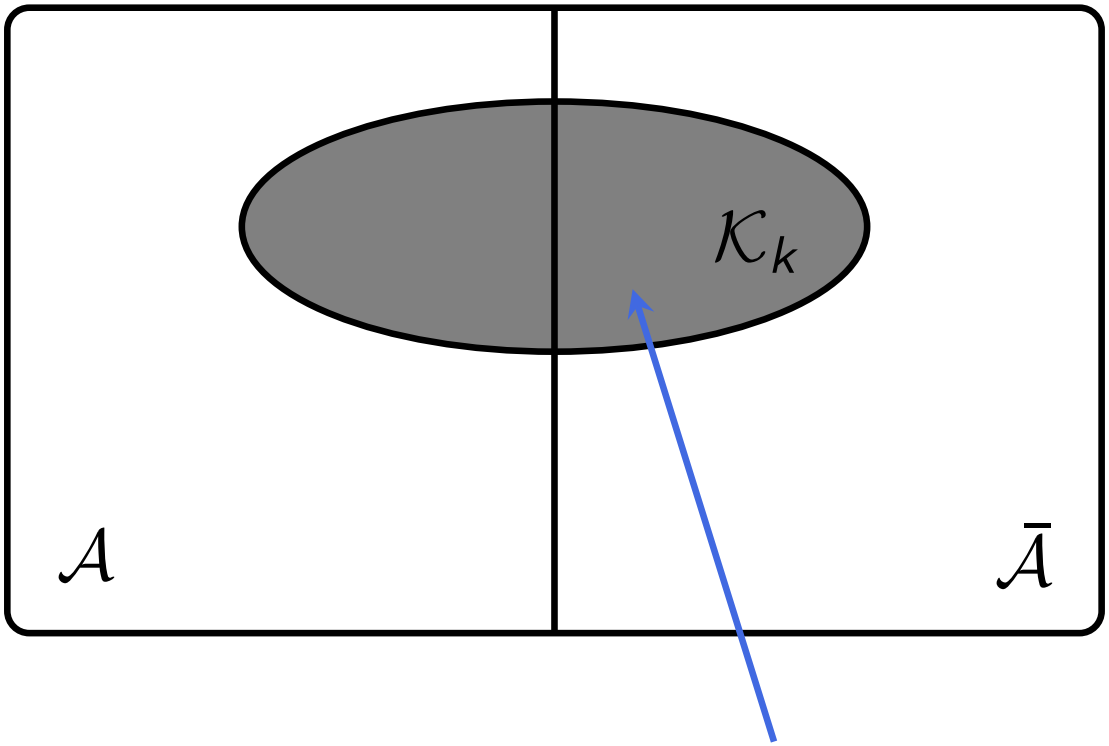
Contributions



- MT-LE: Message-Terminating Leader Election
- PT-LE: Process-Terminating Leader Election

- $\mathcal{A}$: Rings with asymmetric labelling
- $\overline{\mathcal{A}}$: Rings with symmetric labelling
- $\mathcal{U}^*$: Rings with at least one unique label
- $\mathcal{K}_k$: Rings with no more than k processes with the same label

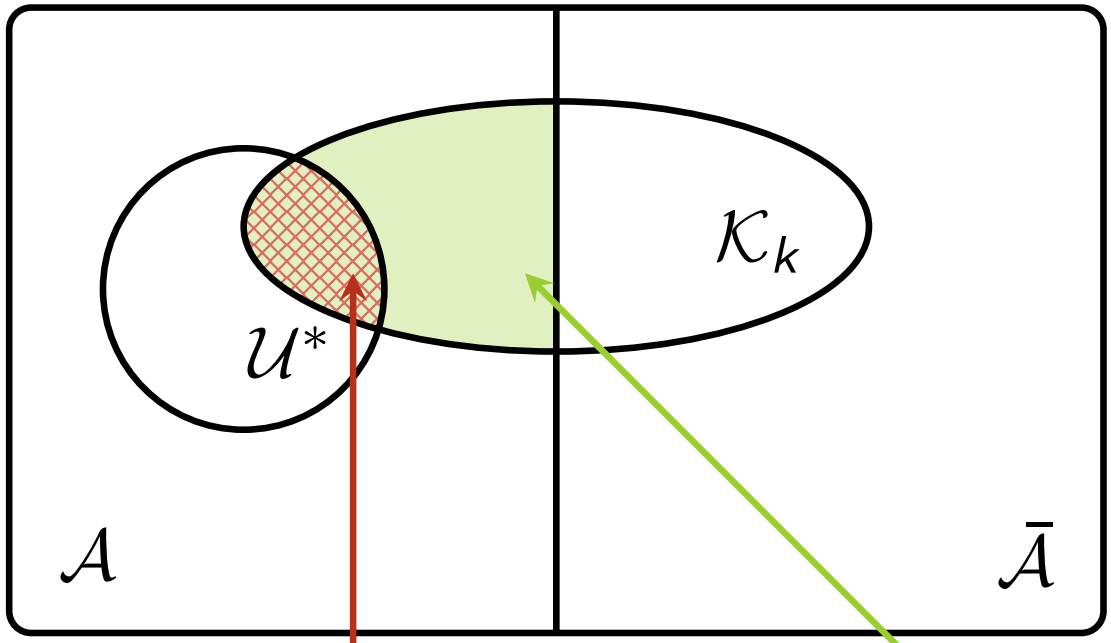
Contributions



- MT-LE: Message-Terminating Leader Election
- PT-LE: Process-Terminating Leader Election

- $\mathcal{A}$: Rings with asymmetric labelling
- $\overline{\mathcal{A}}$: Rings with symmetric labelling
- $\mathcal{U}^*$: Rings with at least one unique label
- $\mathcal{K}_k$: Rings with no more than k processes with the same label

Contributions



PT-LE Algorithms for $\mathcal{A} \cap \mathcal{K}_k$

PT-LE Algorithm for $\mathcal{U}^* \cap \mathcal{K}_k$

- MT-LE: Message-Terminating Leader Election
- PT-LE: Process-Terminating Leader Election

- $\mathcal{A}$: Rings with asymmetric labelling
- $\overline{\mathcal{A}}$: Rings with symmetric labelling
- $\mathcal{U}^*$: Rings with at least one unique label
- $\mathcal{K}_k$: Rings with no more than k processes with the same label

Lower bound for $\mathcal{U}^* \cap \mathcal{K}_k$

Lower bound

Lemma

Let $k \geq 2$.

Let A *be an algorithm that solves the PT-LE for* $\mathcal{U}^* \cap \mathcal{K}_k$.

$\forall R_n \in \mathcal{K}_1$ *of* n *processes, the synchronous execution of* A *in* R_n *lasts at least* $1 + (k-2)\,n$ *time units.*

Proof Outline (1/3)

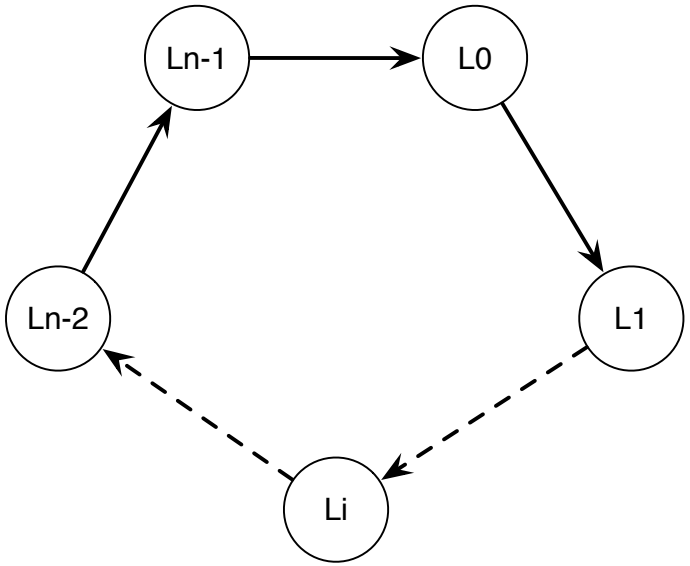


Figure : $R_n \in K_1 \subset \mathcal{U}^* \cap \mathcal{K}_k$

Proof Outline (1/3)

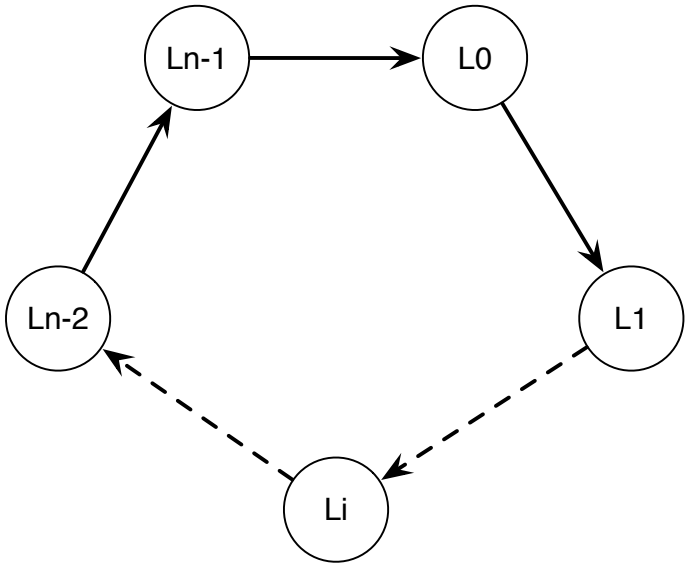


Figure : $R_n \in K_1 \subset \mathcal{U}^* \cap \mathcal{K}_k$

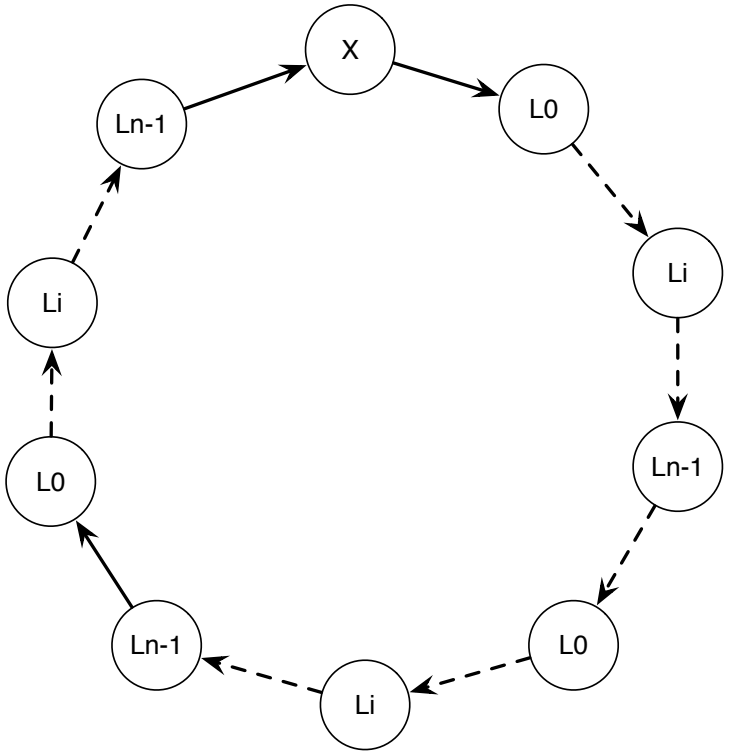


Figure : $R_{n,k} \in \mathcal{U}^* \cap \mathcal{K}_k$

Proof Outline (1/3)

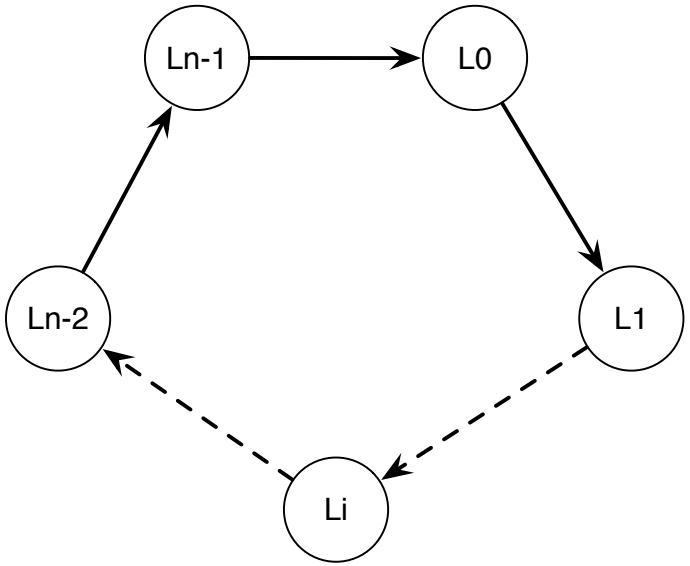


Figure : $R_n \in K_1 \subset \mathcal{U}^* \cap \mathcal{K}_k$

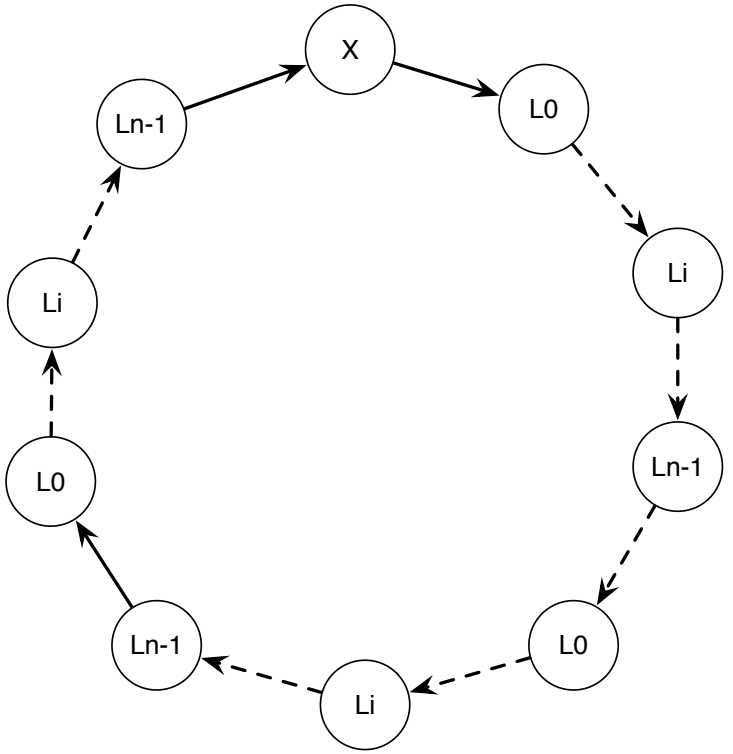


Figure : $R_{n,k} \in \mathcal{U}^* \cap \mathcal{K}_k$

By the contradiction, assume that the synchronous execution of A on R_n terminates before time $1 + (k-2)\,n$.

Proof Outline (2/3)

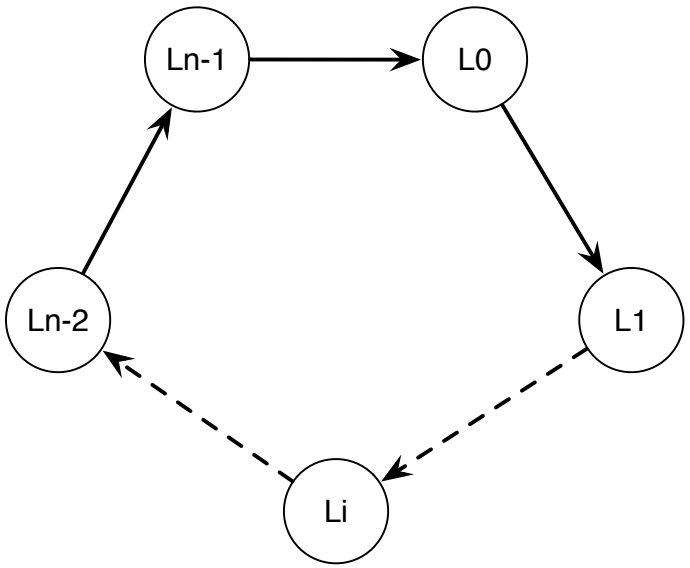


Figure : $R_n \in K_1 \subset \mathcal{U}^* \cap \mathcal{K}_k$

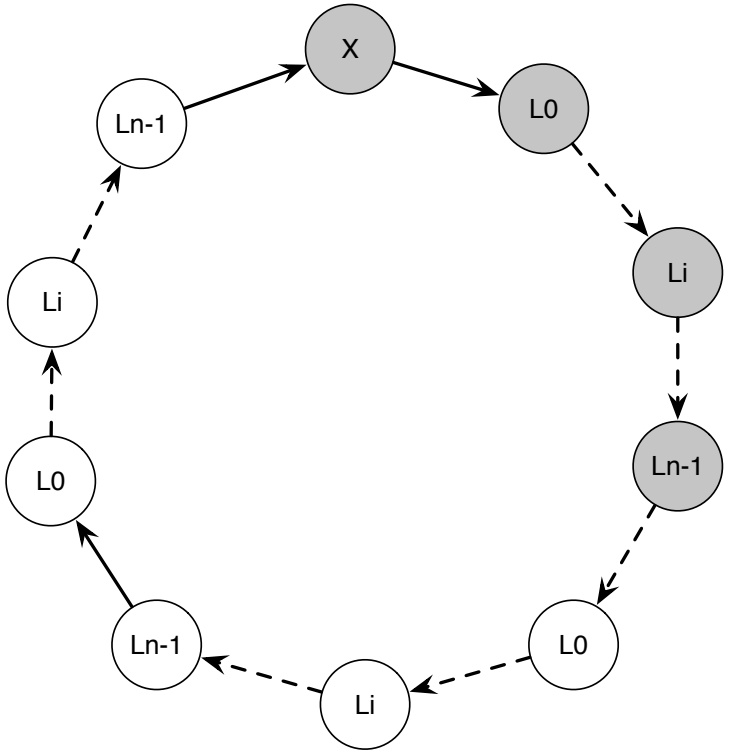


Figure : $R_{n,k} \in \mathcal{U}^* \cap \mathcal{K}_k$

Synchronous execution after up to $T < 1 + (k-2)\,n$ time units.

Proof Outline (3/3)

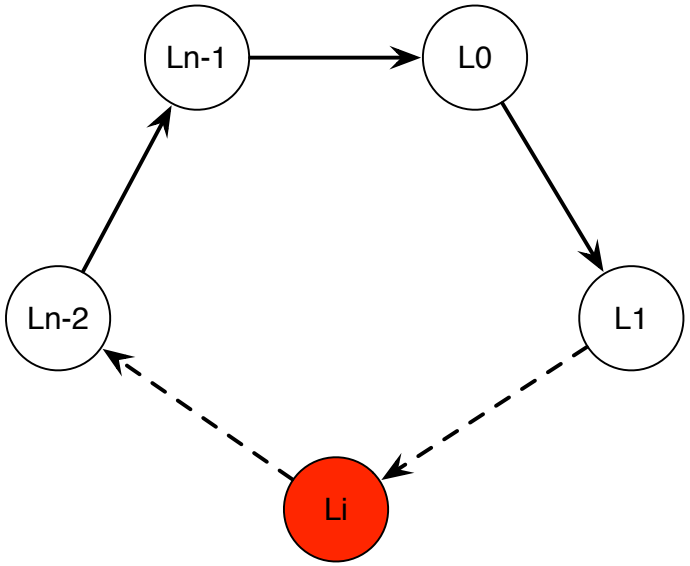


Figure : $R_n \in K_1 \subset \mathcal{U}^* \cap \mathcal{K}_k$

At time T, one node is elected in R_n.

Proof Outline (3/3)

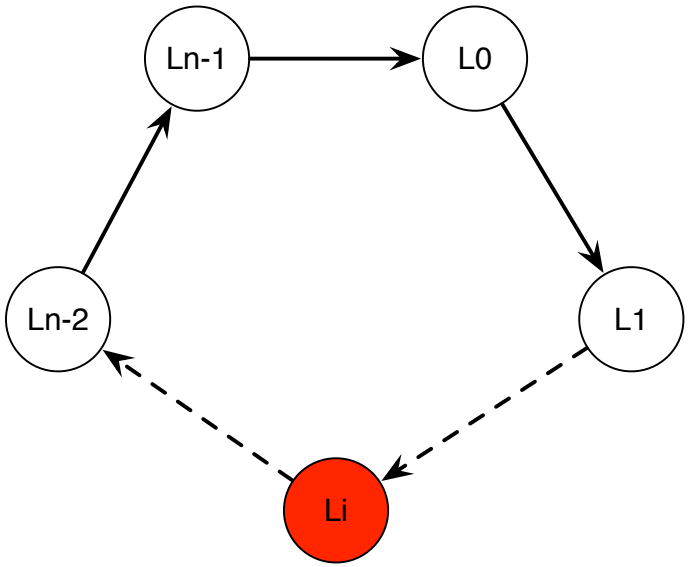


Figure : $R_n \in K_1 \subset \mathcal{U}^* \cap \mathcal{K}_k$

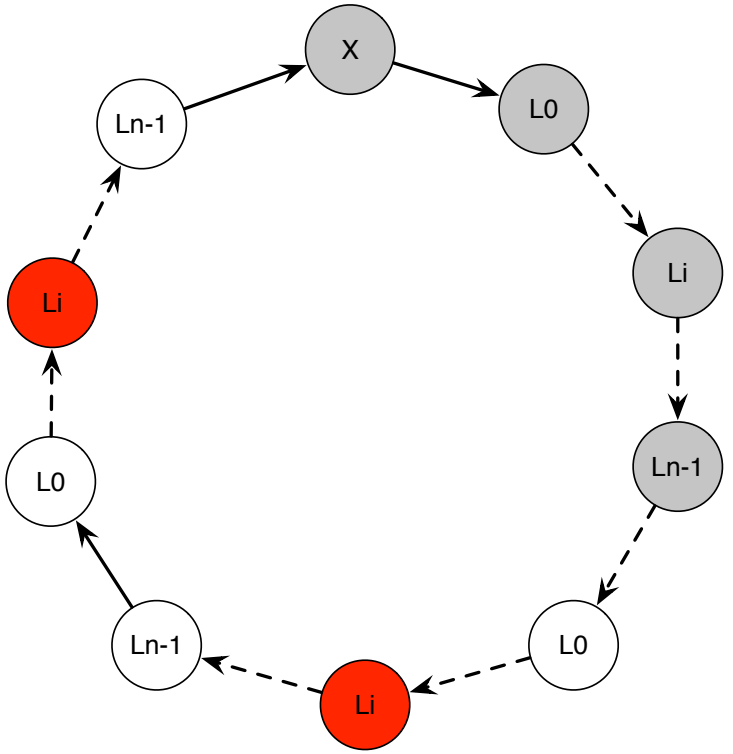


Figure : $R_{n,k} \in \mathcal{U}^* \cap \mathcal{K}_k$

At time T, one node is elected in R_n.

But, two nodes are elected in $R_{n,k}$, contradiction.

Consequences (1/2)

Corollary

Let $k \geq 2$. The time complexity of any algorithm that solves the process-terminating leader election for $\mathcal{U}^* \cap \mathcal{K}_k$ (resp. $\mathcal{A} \cap \mathcal{K}_k$) is $\Omega(k\,n)$ time units, where n is the number of processes.

Consequences (2/2)

Theorem

There is no algorithm that solves the process-terminating leader election for $\mathcal{U}^*$ (resp. $\mathcal{A}$).

Consequences (2/2)

Theorem

There is no algorithm that solves the process-terminating leader election for $\mathcal{U}^*$ (resp. $\mathcal{A}$).

By the contradiction, let A be a PT-LE algorithm for $\mathcal{U}^*$.

Consequences (2/2)

Theorem

There is no algorithm that solves the process-terminating leader election for $\mathcal{U}^*$ (resp. $\mathcal{A}$).

By the contradiction, let A be a PT-LE algorithm for $\mathcal{U}^*$.

By definition, A solves PT-LE in $\mathcal{U}^* \cap \mathcal{K}_3$, $\mathcal{U}^* \cap \mathcal{K}_4$, ...

Let R_n be a ring network of $\mathcal{K}_1$ with n processes.

Consequences (2/2)

Theorem

There is no algorithm that solves the process-terminating leader election for $\mathcal{U}^*$ (resp. $\mathcal{A}$).

By the contradiction, let A be a PT-LE algorithm for $\mathcal{U}^*$.

By definition, A solves PT-LE in $\mathcal{U}^* \cap \mathcal{K}_3$, $\mathcal{U}^* \cap \mathcal{K}_4$, ...

Let R_n be a ring network of $\mathcal{K}_1$ with n processes.

Since $R_n \in \mathcal{U}^* \cap \mathcal{K}_3$, the synchronous execution of A in R_n lasts at least $1 + n$ time units, by Lemma 1.

Consequences (2/2)

Theorem

There is no algorithm that solves the process-terminating leader election for $\mathcal{U}^*$ (resp. $\mathcal{A}$).

By the contradiction, let A be a PT-LE algorithm for $\mathcal{U}^*$.

By definition, A solves PT-LE in $\mathcal{U}^* \cap \mathcal{K}_3$, $\mathcal{U}^* \cap \mathcal{K}_4$, ...

Let R_n be a ring network of $\mathcal{K}_1$ with n processes.

Since $R_n \in \mathcal{U}^* \cap \mathcal{K}_3$, the synchronous execution of A in R_n lasts at least $1 + n$ time units, by Lemma 1.

Since $R_n \in \mathcal{U}^* \cap \mathcal{K}_4$, the synchronous execution of A in R_n lasts at least $1 + 2n$ time units, by Lemma 1.

Consequences (2/2)

Theorem

There is no algorithm that solves the process-terminating leader election for $\mathcal{U}^*$ (resp. $\mathcal{A}$).

By the contradiction, let A be a PT-LE algorithm for $\mathcal{U}^*$.

By definition, A solves PT-LE in $\mathcal{U}^* \cap \mathcal{K}_3$, $\mathcal{U}^* \cap \mathcal{K}_4$, ...

Let R_n be a ring network of $\mathcal{K}_1$ with n processes.

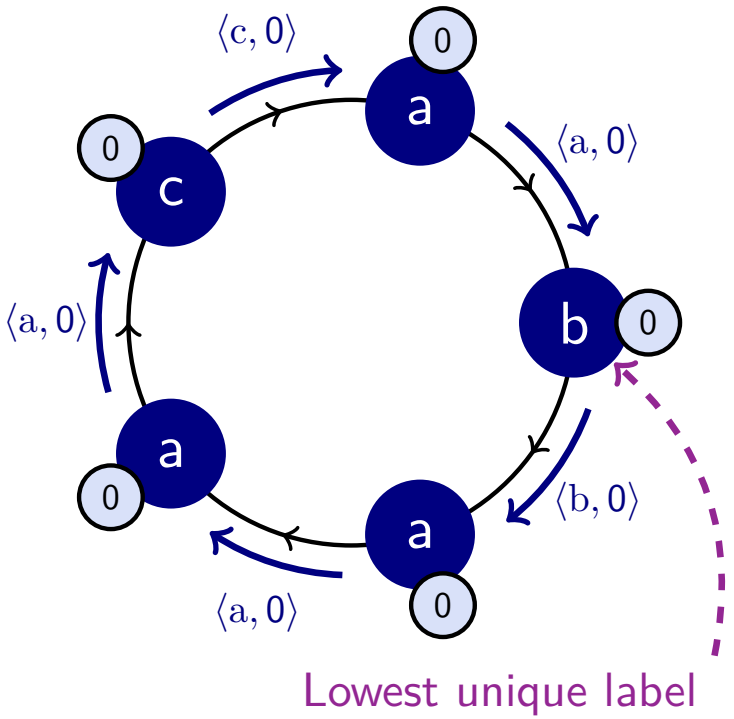
Since $R_n \in \mathcal{U}^* \cap \mathcal{K}_3$, the synchronous execution of A in R_n lasts at least $1 + n$ time units, by Lemma 1.

Since $R_n \in \mathcal{U}^* \cap \mathcal{K}_4$, the synchronous execution of A in R_n lasts at least $1 + 2n$ time units, by Lemma 1.

...

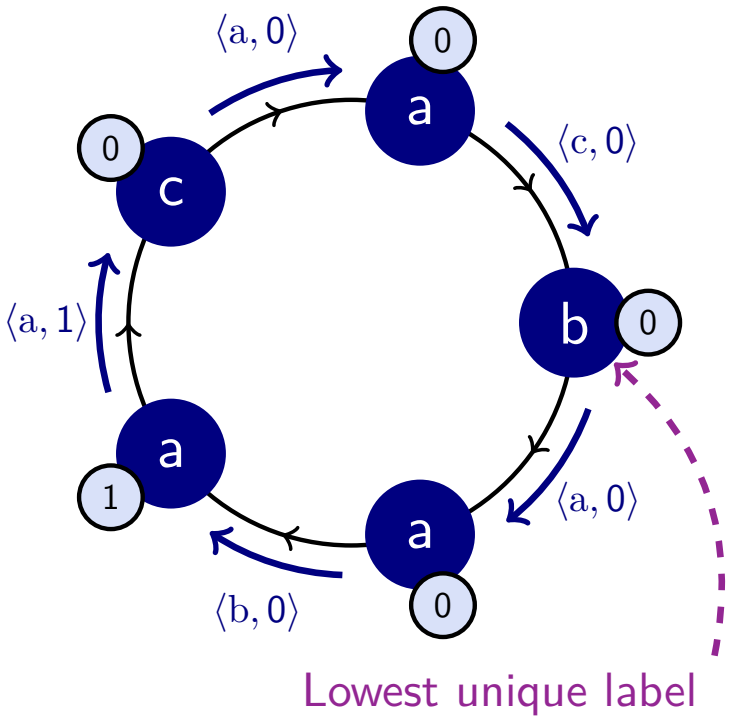
Algorithm for $\mathcal{U}^* \cap \mathcal{K}_k$

PT-LE Algorithm U_k for $\mathcal{U}^* \cap \mathcal{K}_k$



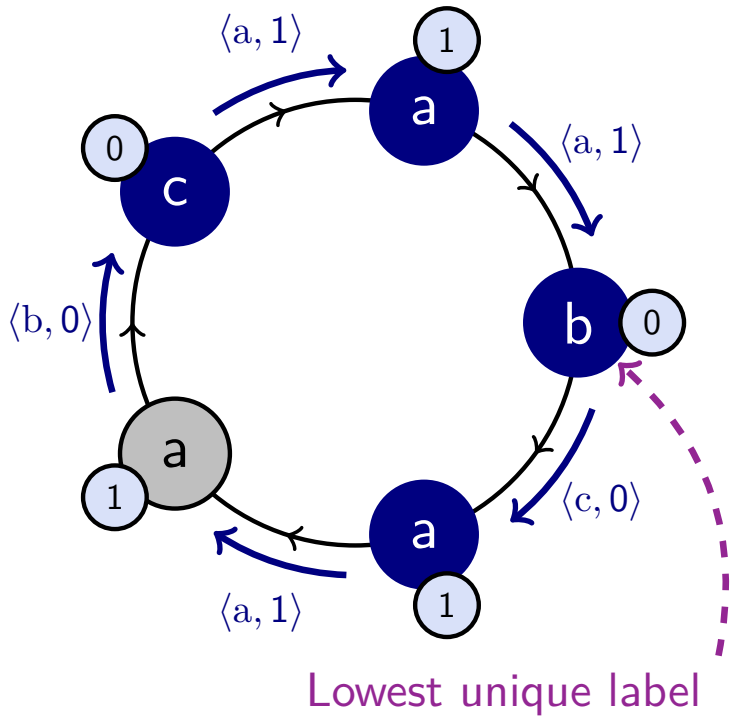
- Counter = rough estimation of the predominance

PT-LE Algorithm U_k for $\mathcal{U}^* \cap \mathcal{K}_k$



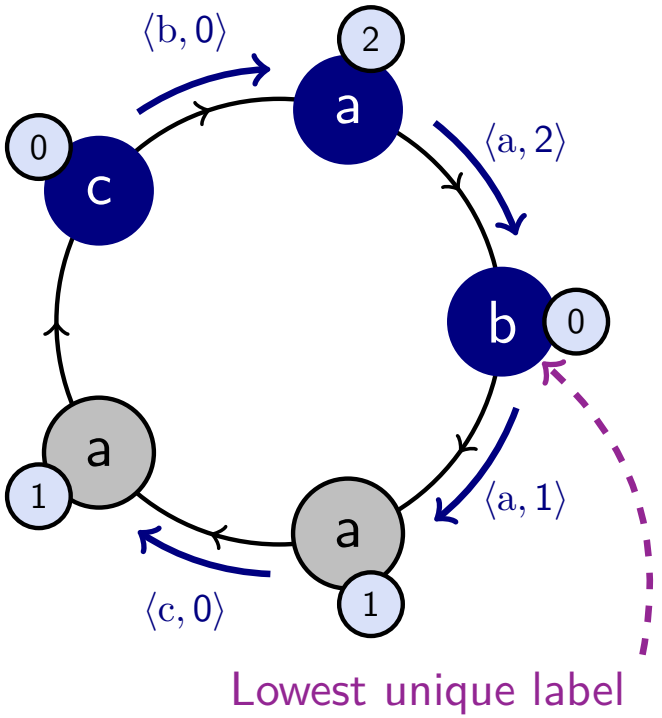
- Counter = rough estimation of the predominance
- Process elimination:
  - Lower counter, $\neq$ label $\rightarrow$ not unique

PT-LE Algorithm U_k for $\mathcal{U}^* \cap \mathcal{K}_k$



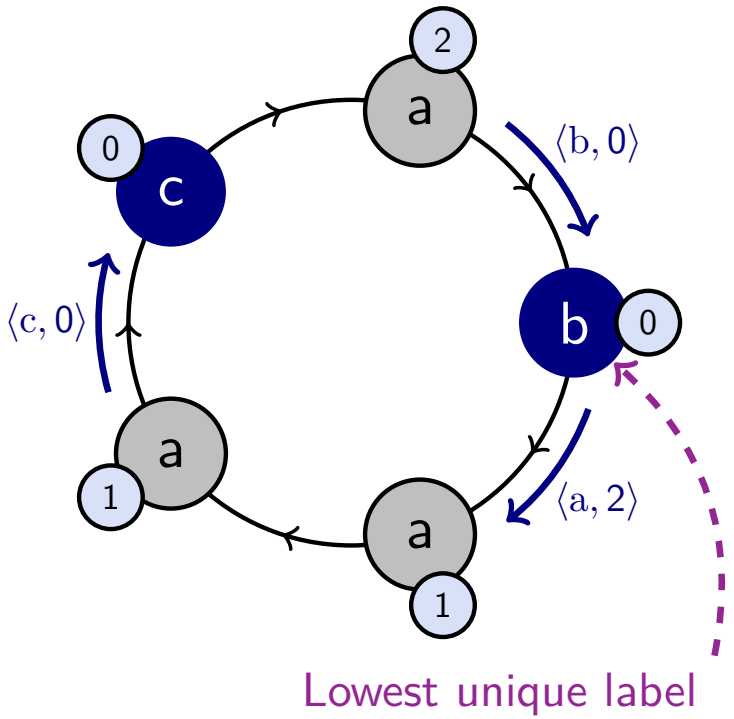
- Counter = rough estimation of the predominance
- Process elimination:
  - Lower counter, $\neq$ label $\rightarrow$ not unique
- Message elimination:
  - Passive, same ID $\rightarrow$ not relevant

PT-LE Algorithm U_k for $\mathcal{U}^* \cap \mathcal{K}_k$



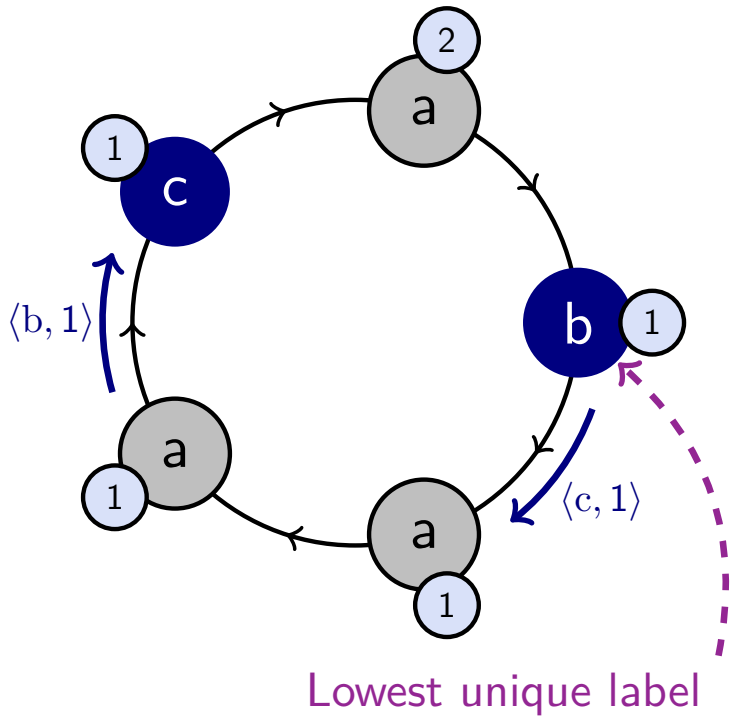
- Counter = rough estimation of the predominance
- Process elimination:
  - Lower counter, $\neq$ label $\rightarrow$ not unique
- Message elimination:
  - Passive, same ID $\rightarrow$ not relevant

PT-LE Algorithm U_k for $\mathcal{U}^* \cap \mathcal{K}_k$



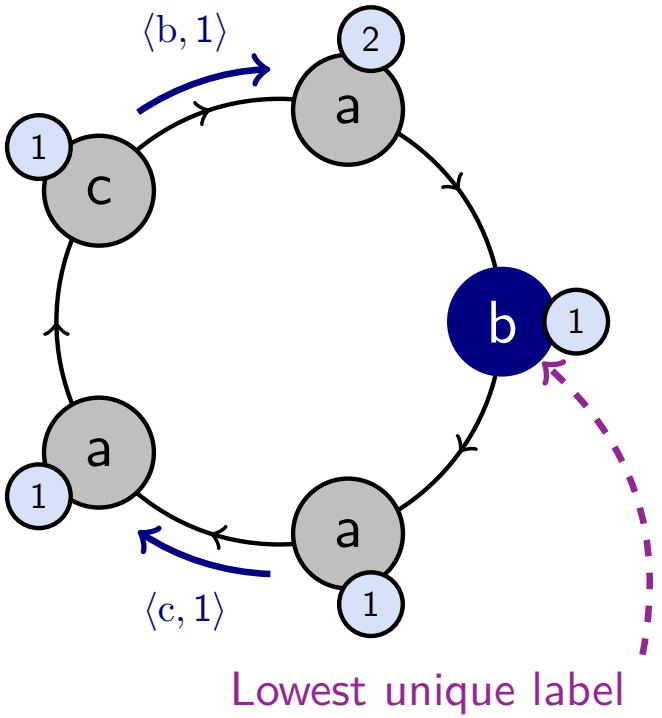
- Counter = rough estimation of the predominance
- Process elimination:
  - Lower counter, $\neq$ label $\rightarrow$ not unique
- Message elimination:
  - Passive, same ID $\rightarrow$ not relevant

PT-LE Algorithm U_k for $\mathcal{U}^* \cap \mathcal{K}_k$



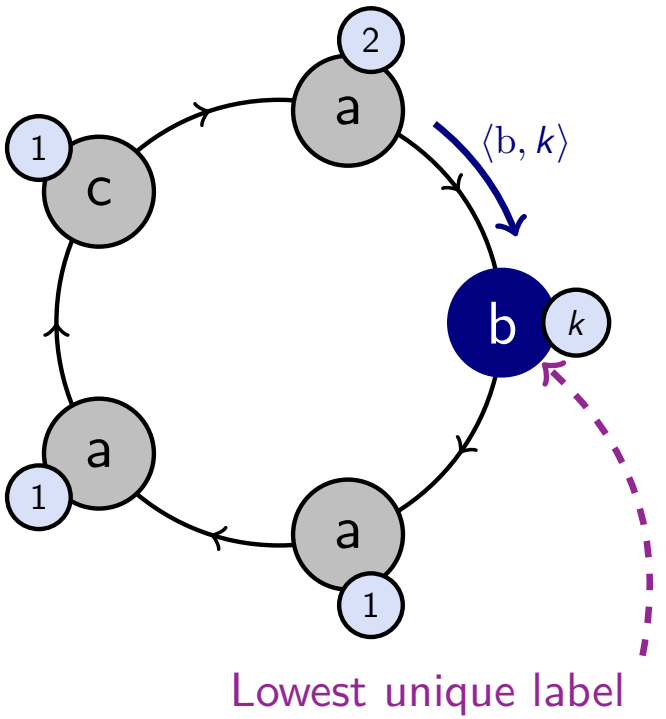
- Counter = rough estimation of the predominance
- Process elimination:
  - Lower counter, $\neq$ label $\rightarrow$ not unique
  - Same counter $\neq 0$, lower label $\rightarrow$ not lowest unique
- Message elimination:
  - Passive, same ID $\rightarrow$ not relevant

PT-LE Algorithm U_k for $\mathcal{U}^* \cap \mathcal{K}_k$



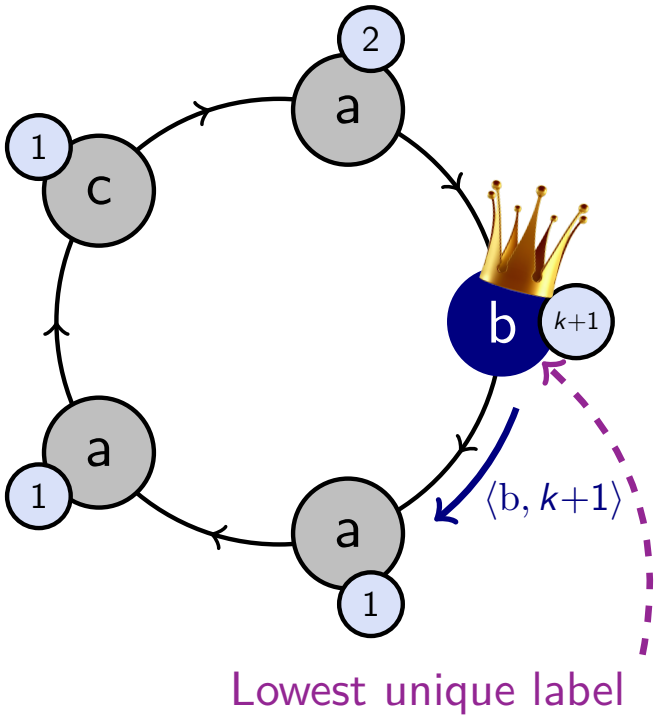
- Counter = rough estimation of the predominance
- Process elimination:
  - Lower counter, $\neq$ label $\rightarrow$ not unique
  - Same counter $\neq 0$, lower label $\rightarrow$ not lowest unique
- Message elimination:
  - Passive, same ID $\rightarrow$ not relevant

PT-LE Algorithm U_k for $\mathcal{U}^* \cap \mathcal{K}_k$



- Counter = rough estimation of the predominance
- Process elimination:
  - Lower counter, $\neq$ label $\rightarrow$ not unique
  - Same counter $\neq 0$, lower label $\rightarrow$ not lowest unique
- Message elimination:
  - Passive, same ID $\rightarrow$ not relevant
- Phases:
  - 1st traversal: no more active non-unique labels
  - 2nd traversal: no more active non-lowest unique labels
  - Election detection: receiving $\langle id, k \rangle$

PT-LE Algorithm U_k for $\mathcal{U}^* \cap \mathcal{K}_k$



- Counter = rough estimation of the predominance
- Process elimination:
  - Lower counter, $\neq$ label $\rightarrow$ not unique
  - Same counter $\neq 0$, lower label $\rightarrow$ not lowest unique
- Message elimination:
  - Passive, same ID $\rightarrow$ not relevant
- Phases:
  - 1st traversal: no more active non-unique labels
  - 2nd traversal: no more active non-lowest unique labels
  - Election detection: receiving $\langle id, k \rangle$

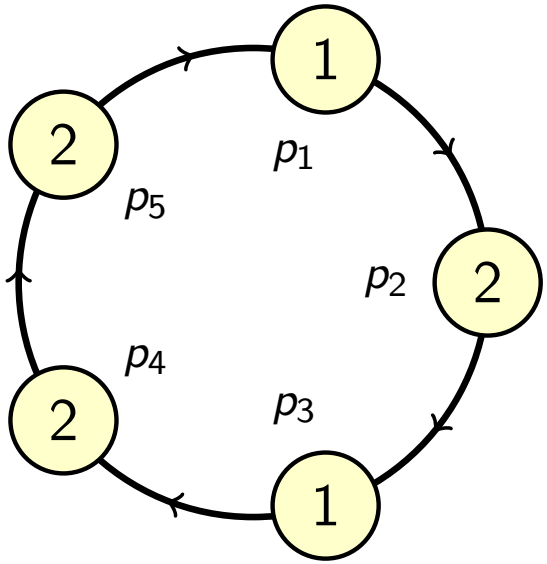
PT-LE Algorithm for $\mathcal{U}^* \cap \mathcal{K}_k$

- **Time complexity:** at most $n(k+2)$
  Asymptotically optimal (work under submission)
- **# messages:** $O(n^2 + kn)$
- **Memory requirement:** $\lceil \log(k+1) \rceil + \log(n) + 4$

Algorithms for $\mathcal{A} \cap \mathcal{K}_k$

A_k, first PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

- **Chosen Leader:**
  process whose LabelSequence = LyndonWord(LabelSequence)
  Lyndon Word = smallest rotation in lexicographic order

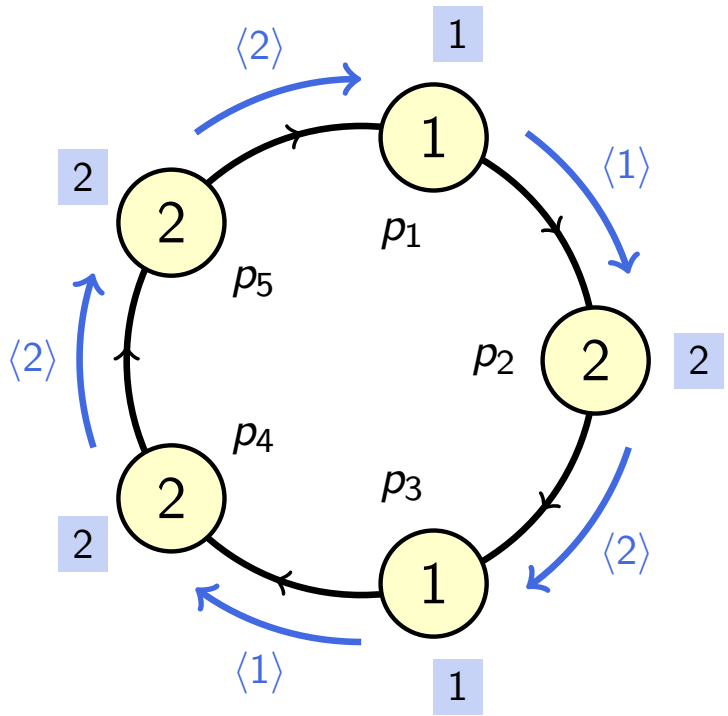


- **Label Sequence at p_1:**
  $LS_{p_1} = 12212$
  Rotations:

12212	$(= LS_{p_1})$	
21221	$(= LS_{p_2})$	
12122	$(= LS_{p_3})$	$LW \neq LS_{p_1}$
21212	$(= LS_{p_4})$	
22121	$(= LS_{p_5})$	

A_k, first PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

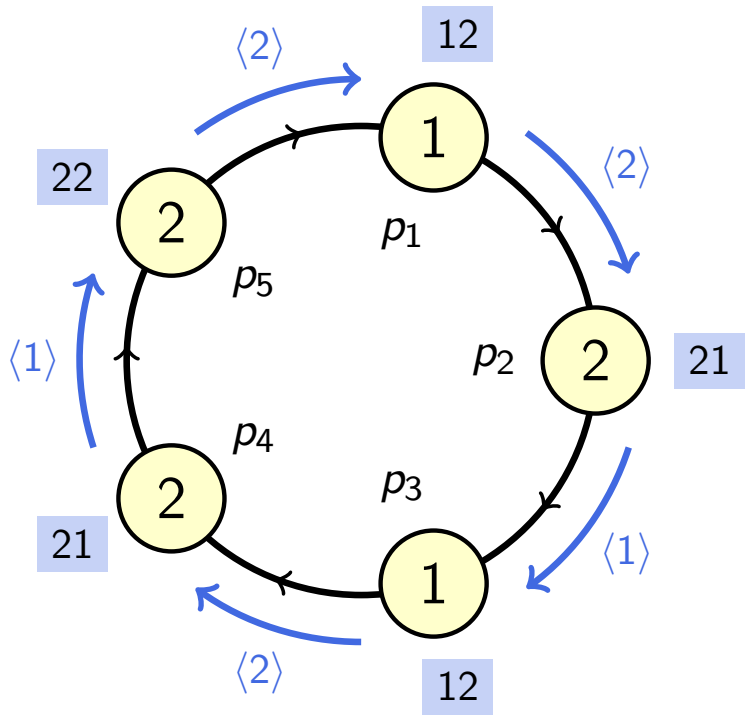
- **Chosen Leader:**
  process whose LabelSequence = LyndonWord(LabelSequence)
  Lyndon Word = smallest rotation in lexicographic order



- Local label aggregation

A_k, first PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

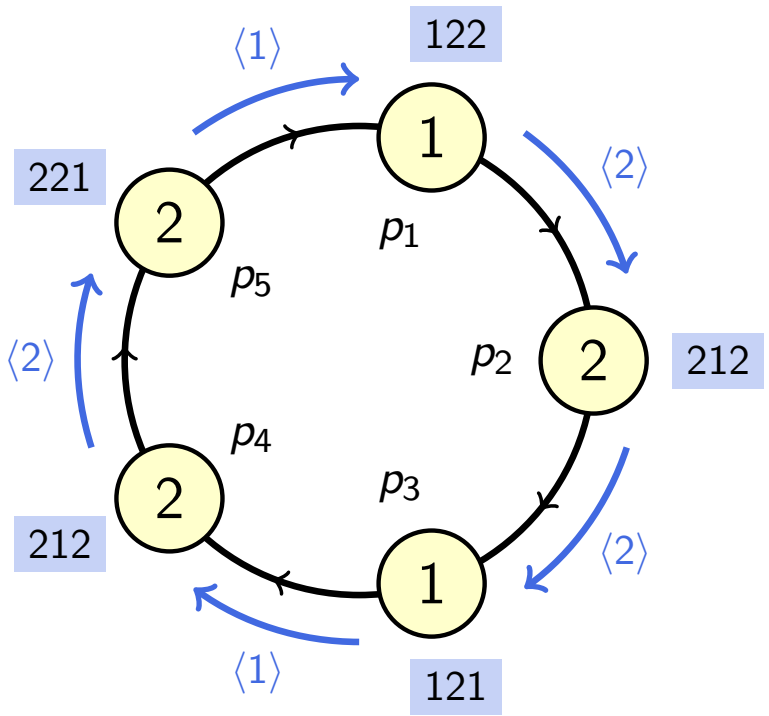
- **Chosen Leader:**
  process whose LabelSequence = LyndonWord(LabelSequence)
  Lyndon Word = smallest rotation in lexicographic order



- Local label aggregation

A_k, first PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

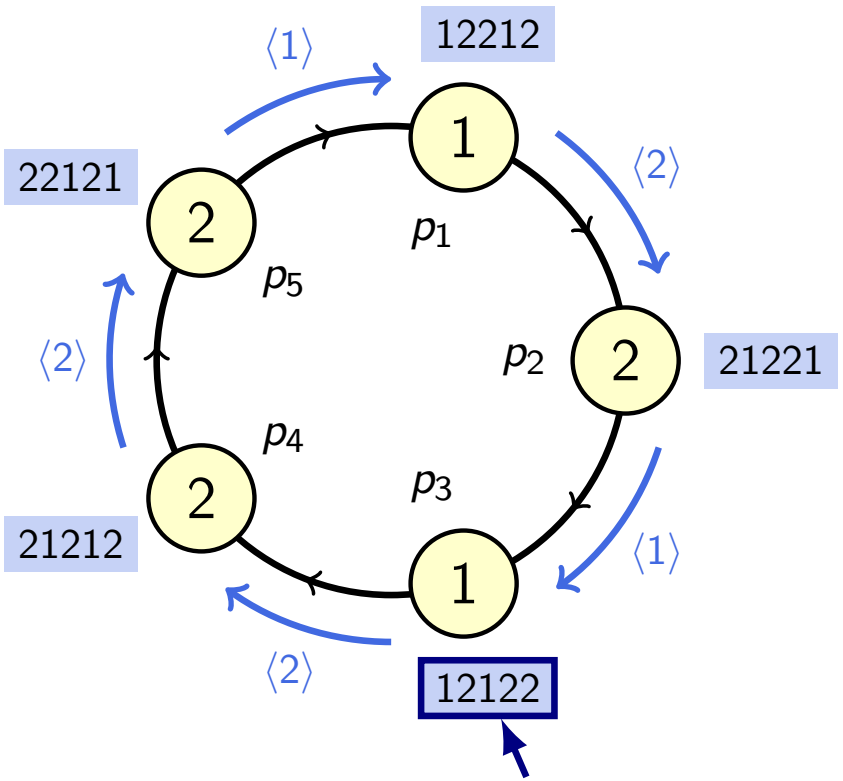
- **Chosen Leader:**
  process whose LabelSequence = LyndonWord(LabelSequence)
  Lyndon Word = smallest rotation in lexicographic order



- Local label aggregation

A_k, first PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

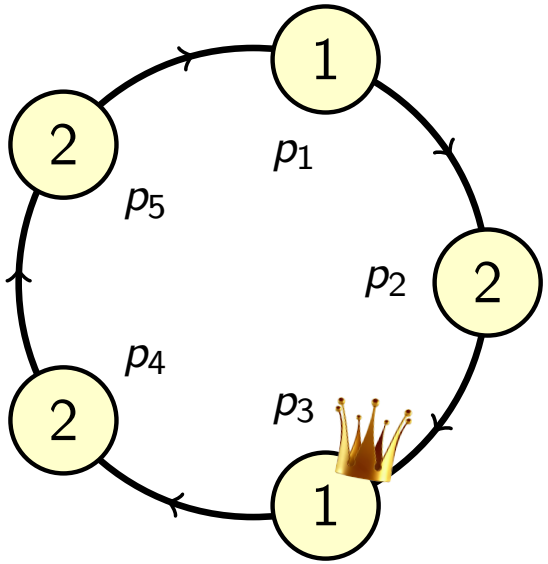
- **Chosen Leader:**
  process whose LabelSequence = LyndonWord(LabelSequence)
  Lyndon Word = smallest rotation in lexicographic order



- Local label aggregation
- ⊖ Do not know n
  $\Rightarrow$ Leader cannot detect its election

A_k, first PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

- **Chosen Leader:**
  process whose LabelSequence = LyndonWord(LabelSequence)
  Lyndon Word = smallest rotation in lexicographic order



$k = 3$

12122 1212212

Smallest repeating prefix = LabelSequence
= LyndonWord(Smallest repeating prefix)

- Local label aggregation
- ⊖ Do not know n
  $\Rightarrow$ Leader cannot detect its election
- Termination detection = $(2k+1) \times$ the same label
  $\Rightarrow$ at least 2 times the sequence of labels

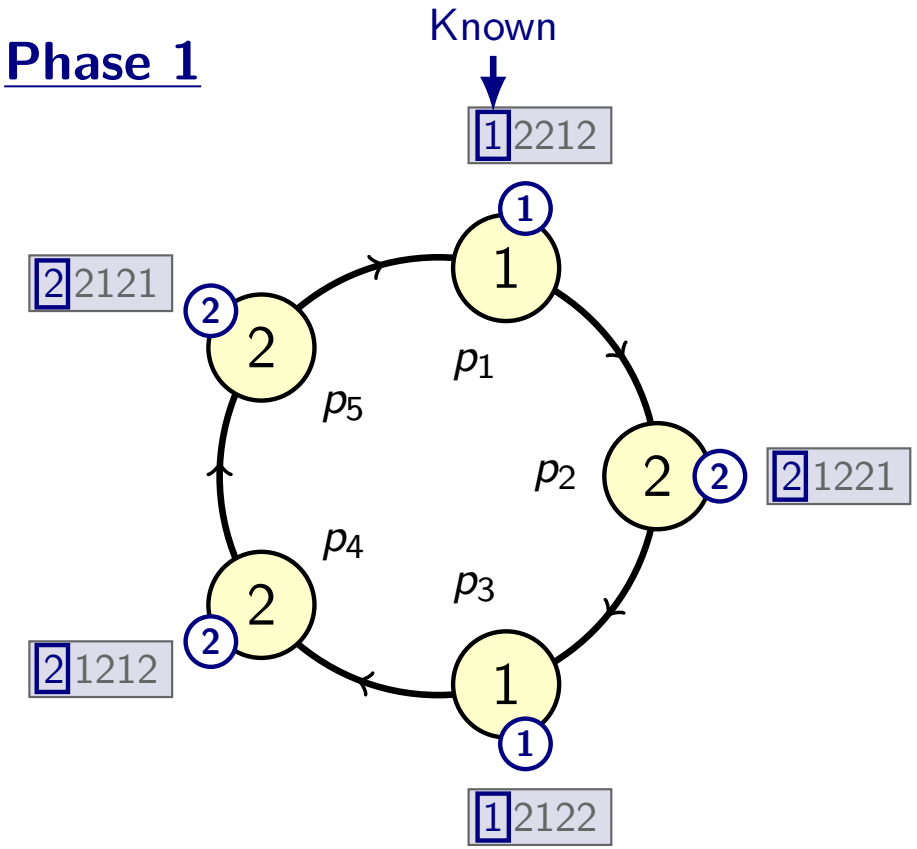
A_k, first PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

- **Time complexity:** at most $(2k+2)n$ time units
- **Message complexity:** at most $n^2(2k+1)$ messages
- **Memory:** $(2k+1)nb + 2b + 3$ bits,
  where b = number of bits to store an ID

Asymptotically optimal time complexity
but
Large memory requirement

B_k, second PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

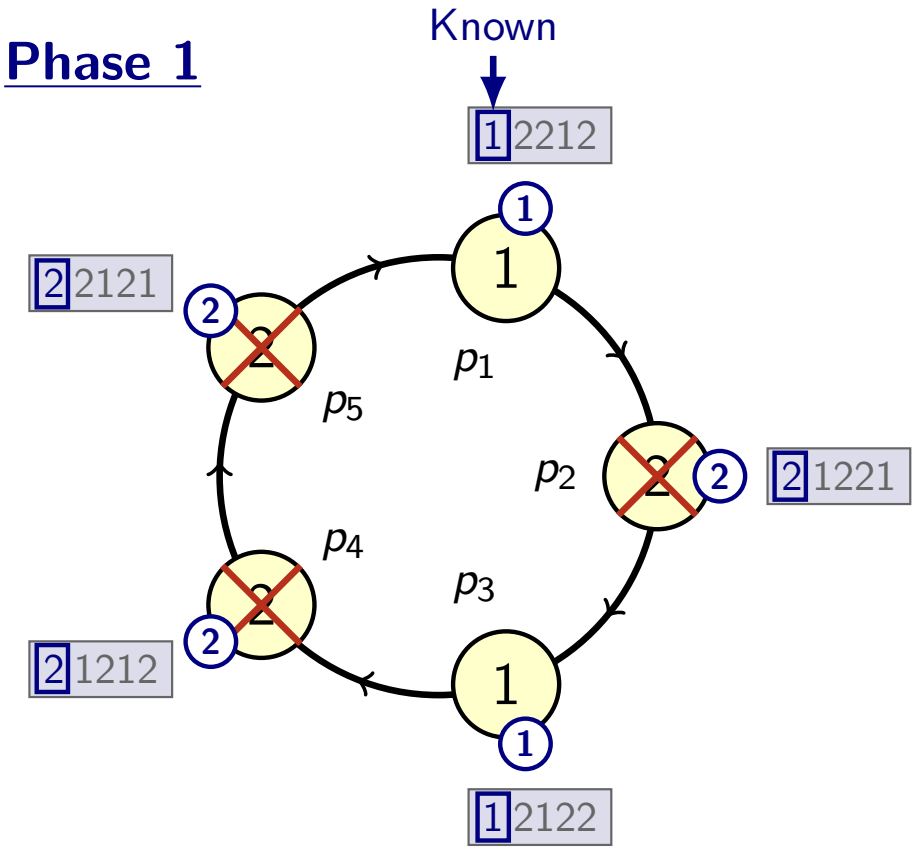
- **Decrease memory usage** ⇒ Peterson principle with radix sort



- During a phase, Known values of active processes circulate clockwise
- End of phase: each still active process received its Known value $k+1$ times

B_k, second PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

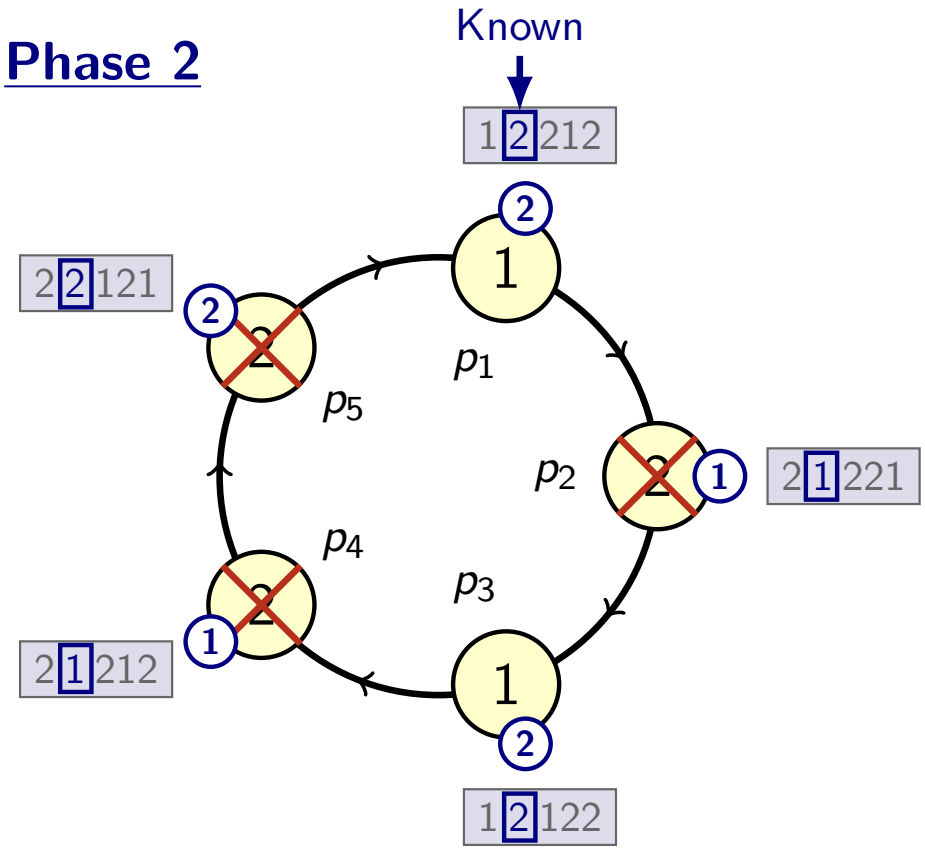
- **Decrease memory usage** $\Rightarrow$ Peterson principle with radix sort



- During a phase, Known values of active processes circulate clockwise
- End of phase: each still active process received its Known value $k+1$ times

B_k, second PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

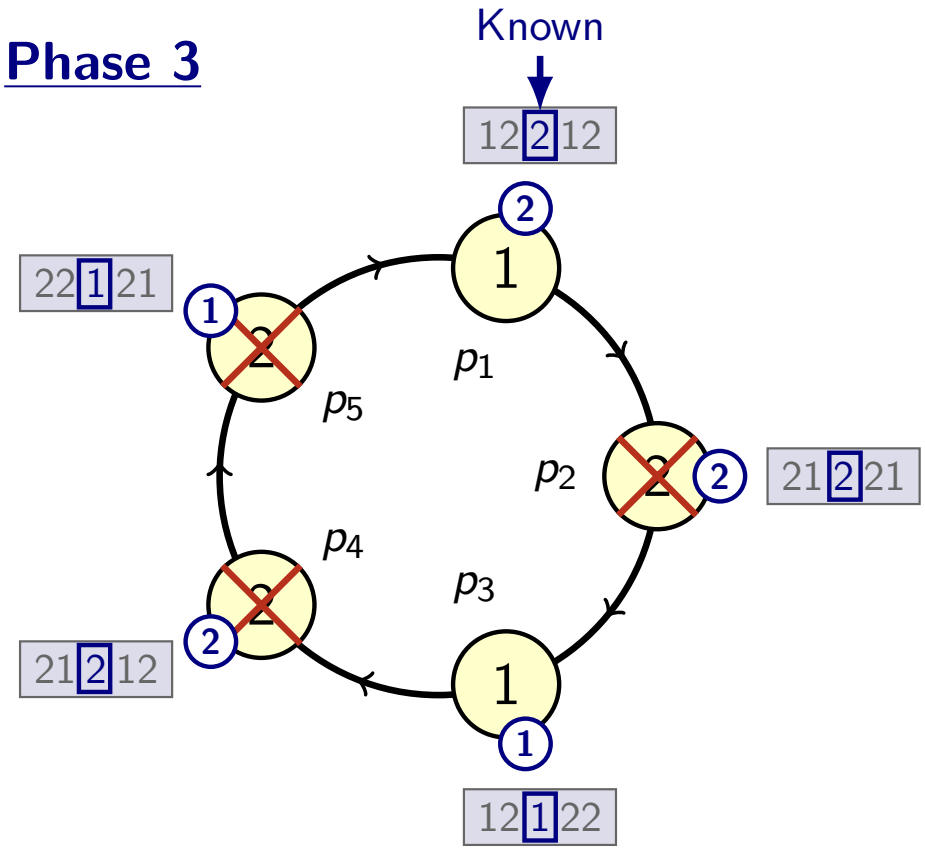
- **Decrease memory usage** $\Rightarrow$ Peterson principle with radix sort



- During a phase, Known values of active processes circulate clockwise
- End of phase: each still active process received its Known value $k+1$ times

B_k, second PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

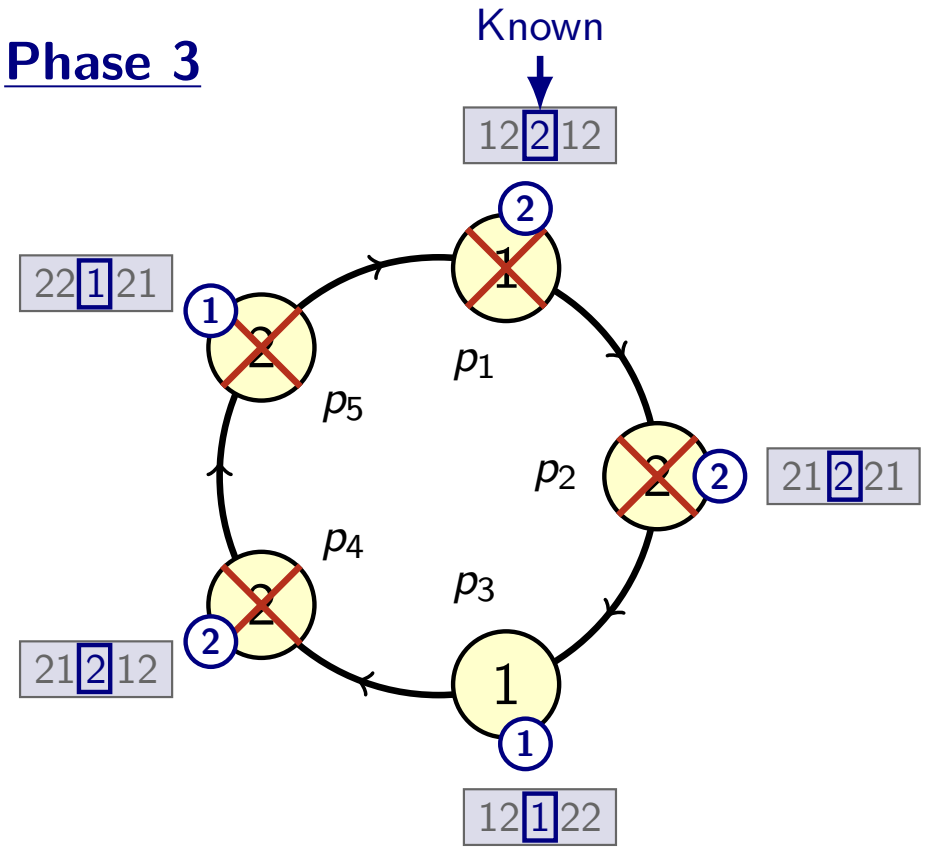
- **Decrease memory usage** $\Rightarrow$ Peterson principle with radix sort



- During a phase, Known values of active processes circulate clockwise
- End of phase: each still active process received its Known value $k+1$ times

B_k, second PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

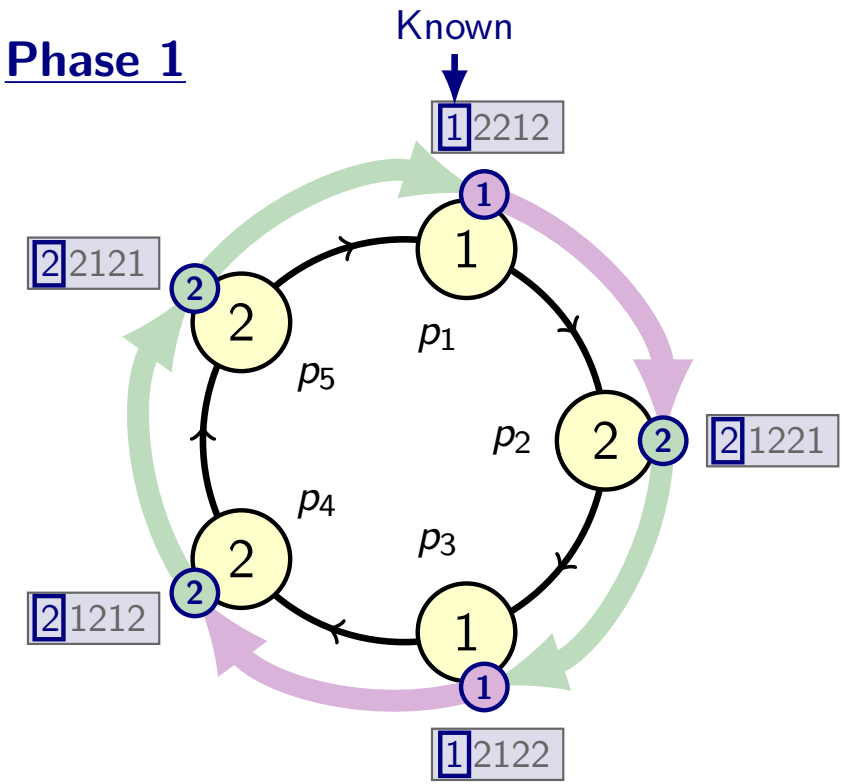
- **Decrease memory usage** ⇒ Peterson principle with radix sort

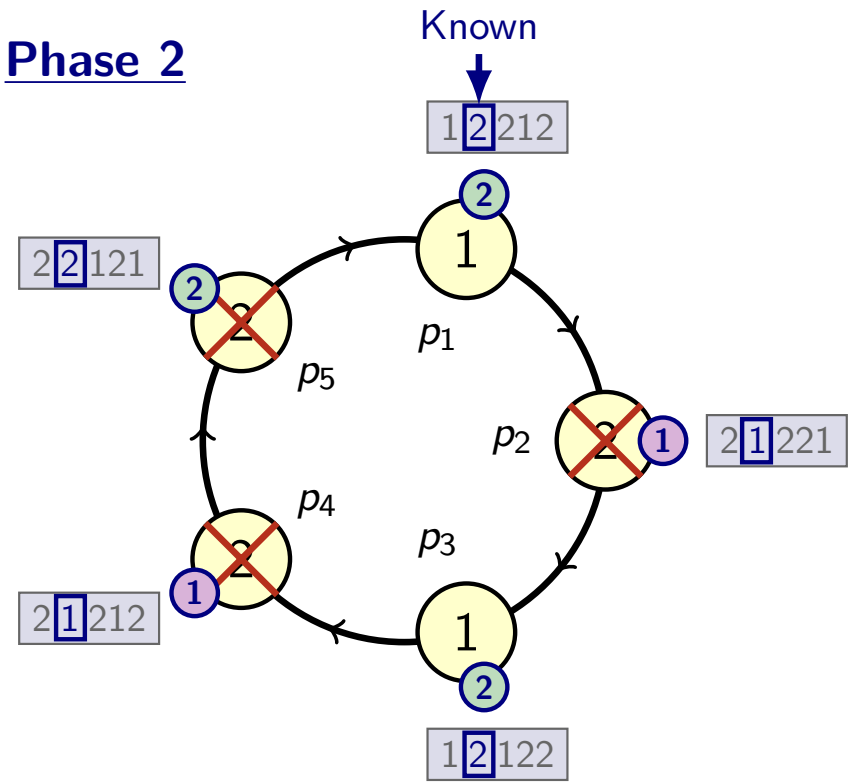


- During a phase, Known values of active processes circulate clockwise
- End of phase: each still active process received its Known value $k+1$ times

B_k, second PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

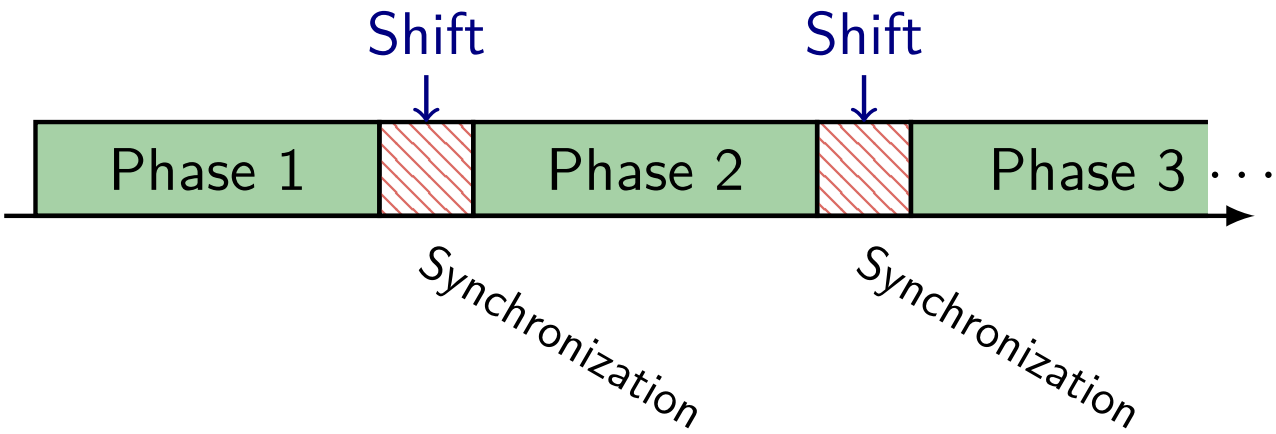
- **Phase Shift**





B_k, second PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

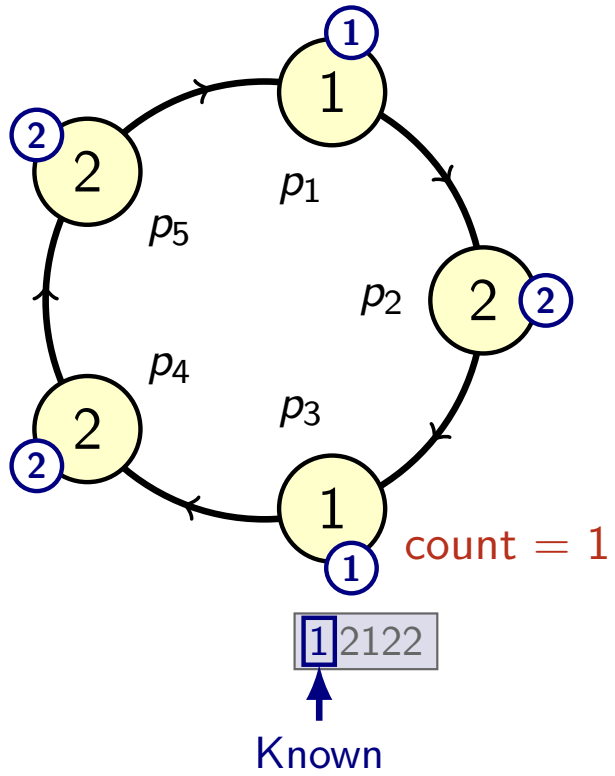
- **Execution**



B_k, second PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

- **Termination Detection:** count = k+1
  count = # phases where Known = Label

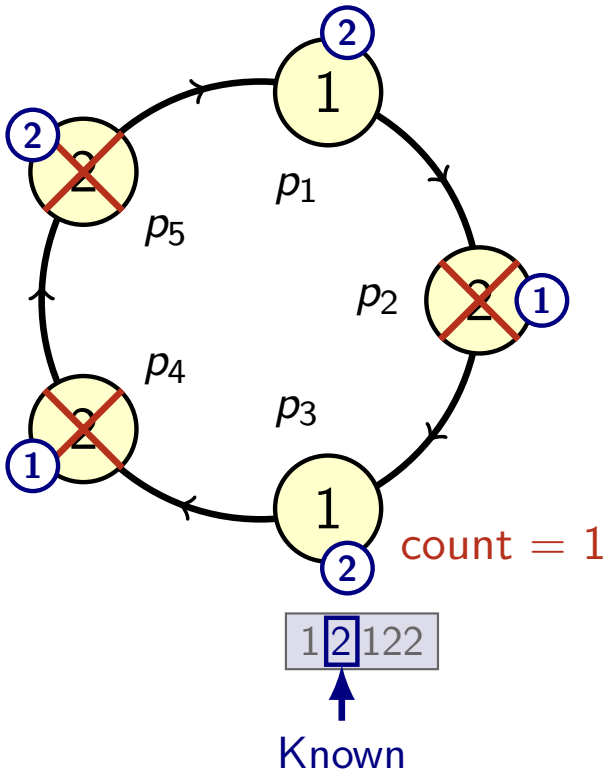
Phase 1



B_k, second PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

- **Termination Detection:** count = k+1
  count = # phases where Known = Label

Phase 2

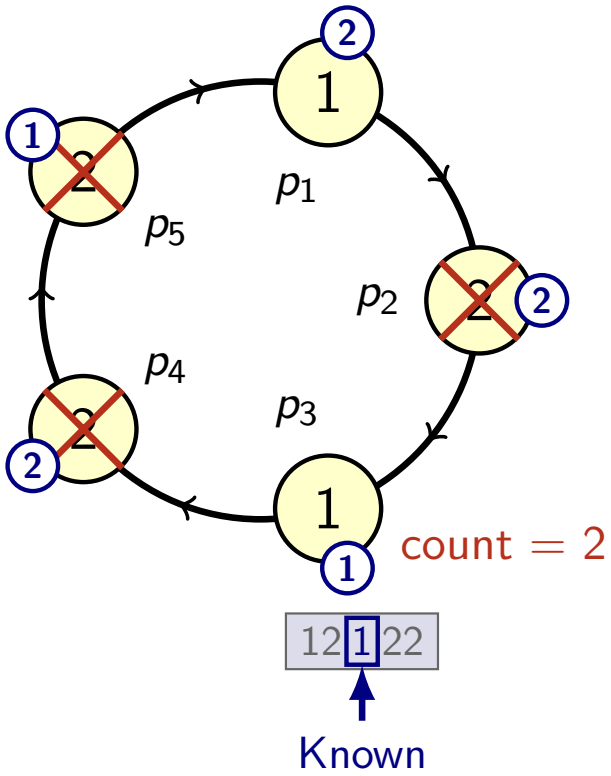


B_k, second PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

- **Termination Detection:** count = k+1

  count = # phases where Known = Label

Phase 3

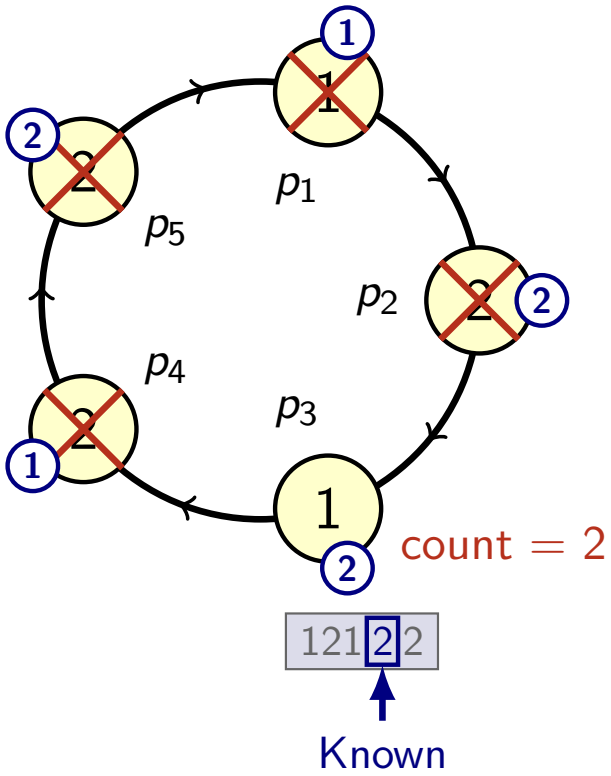


B_k, second PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

- **Termination Detection:** count = k+1

  count = # phases where Known = Label

Phase 4

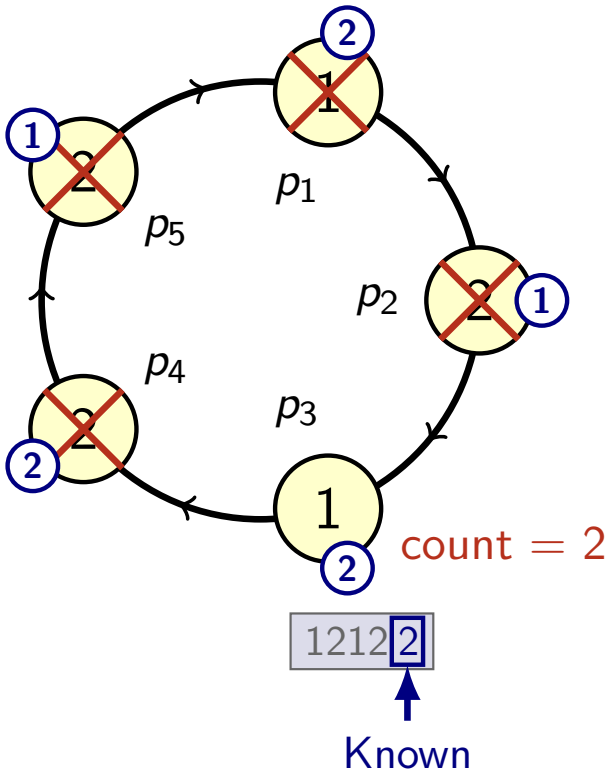


B_k, second PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

- **Termination Detection:** count = k+1

  count = # phases where Known = Label

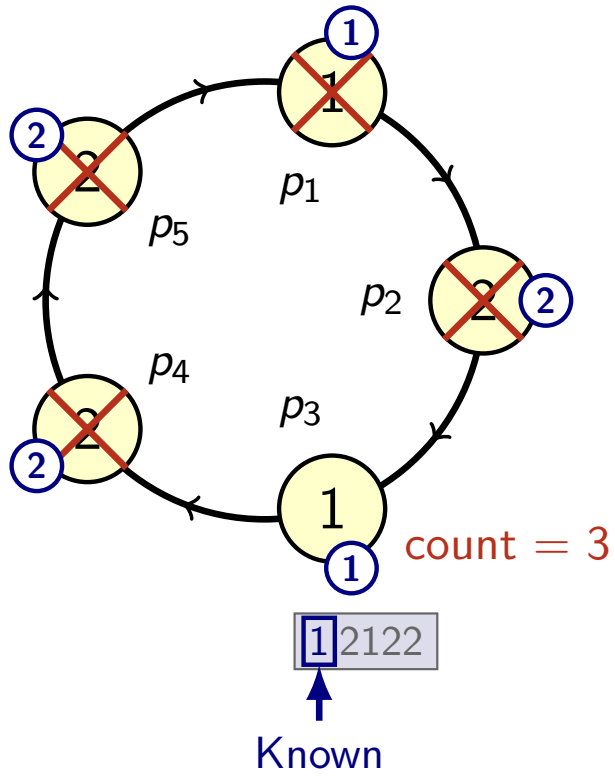
Phase 5



B_k, second PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

- **Termination Detection:** count = k+1
  count = # phases where Known = Label

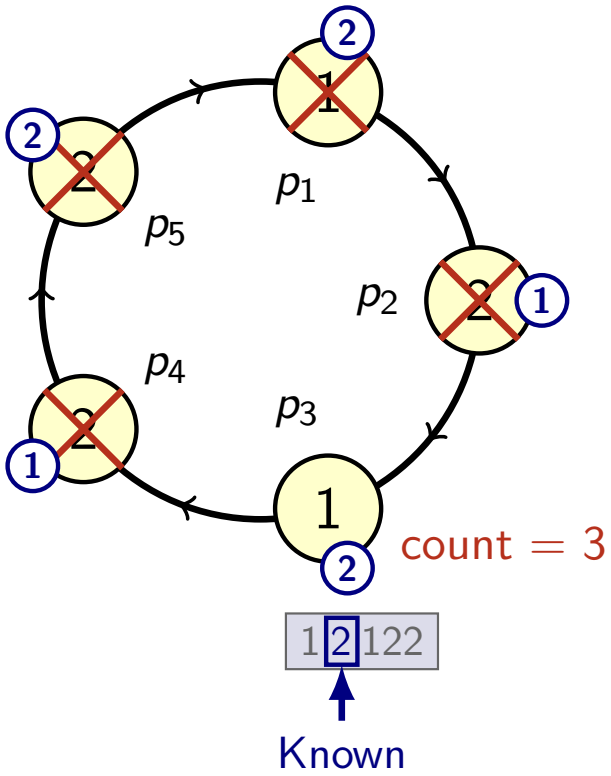
Phase 6



B_k, second PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

- **Termination Detection:** count = k+1
  count = # phases where Known = Label

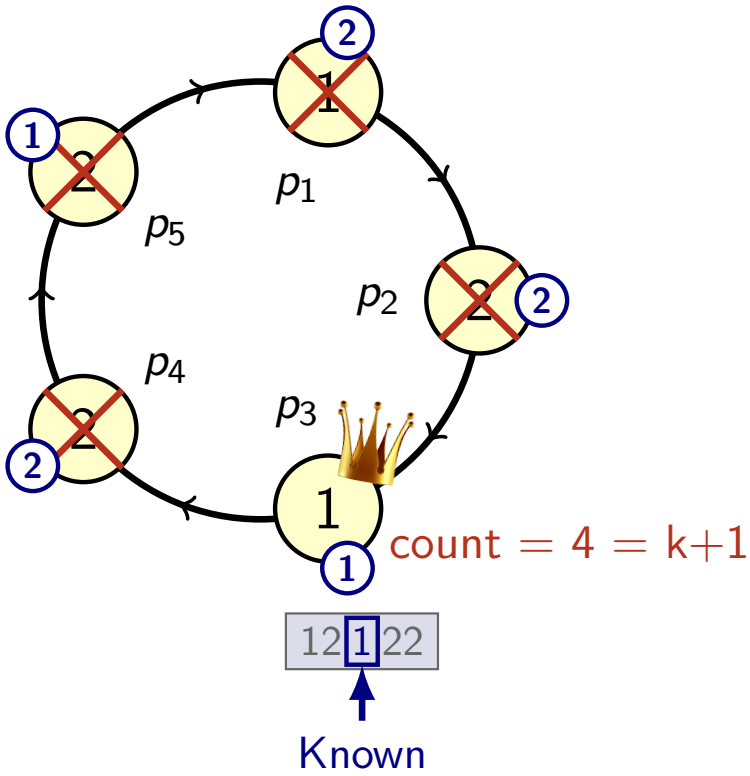
Phase 7



B_k, second PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

- **Termination Detection:** count = k+1
  count = # phases where Known = Label

Phase 8



B_k, second PT-LE Algorithm for $\mathcal{A} \cap \mathcal{K}_k$

- **Memory:** $2\lceil \log k \rceil + 3b + 5$ bits,
  where b = number of bits to store an ID
- **Time complexity:** $O(k^2 n^2)$ time units
- **Message complexity:** $O(k^2 n^2)$ messages

Asymptotically optimal memory requirement

but

Large time complexity

Conclusion

Contributions Summary

- **Impossibility results:** $\bar{\mathcal{A}}$, $\mathcal{A}$, $\mathcal{U}^*$, and $\mathcal{K}_k$
- **Lower bounds:**
  - on the time in $\mathcal{U}^* \cap \mathcal{K}_k$ and $\mathcal{A} \cap \mathcal{K}_k$: $\Omega(kn)$
  - on the # bits exchanged in $\mathcal{U}^* \cap \mathcal{K}_k$ and $\mathcal{A} \cap \mathcal{K}_k$: $\Omega(n^2 + kn)$
- **Algorithms:**

	U_k	A_k	B_k
Rings	$\mathcal{U}^* \cap \mathcal{K}_k$	$\mathcal{A} \cap \mathcal{K}_k$	
Time	$O(kn)$	$O(kn)$	$O(k^2n^2)$
# Messages	$O(kn)$	$O(n^2 + kn)$	$O(k^2n^2)$
Bits/process	$O(\log k + b)$	$O(knb)$	$O(\log k + b)$

Key: asymptotically optimal

Perspectives

Perspectives

- Leader election possible in $\mathcal{A} \cap \mathcal{K}_k$ $\mathcal{A}$, but impossible in $\mathcal{A}$: where is the boundary ?

Perspectives

- Leader election possible in $\mathcal{A} \cap \mathcal{K}_k$ $\mathcal{A}$, but impossible in $\mathcal{A}$: where is the boundary ?
- Find a best trade-off leader election algorithm for $\mathcal{A} \cap \mathcal{K}_k$

Perspectives

- Leader election possible in $\mathcal{A} \cap \mathcal{K}_k$ $\mathcal{A}$, but impossible in $\mathcal{A}$: where is the boundary ?
- Find a best trade-off leader election algorithm for $\mathcal{A} \cap \mathcal{K}_k$
- In $\mathcal{A}$, the knowledge of k and n is computationaly equivalent. Is-it still true in bidirectional rings? What about time complexity?

Perspectives

- Leader election possible in $\mathcal{A} \cap \mathcal{K}_k$ $\mathcal{A}$, but impossible in $\mathcal{A}$: where is the boundary ?
- Find a best trade-off leader election algorithm for $\mathcal{A} \cap \mathcal{K}_k$
- In $\mathcal{A}$, the knowledge of k and n is computationaly equivalent. Is-it still true in bidirectional rings? What about time complexity?
- Self-stabilizing leader election in $\mathcal{U}^* \cap \mathcal{K}_k$ and $\mathcal{A} \cap \mathcal{K}_k$. (research line: adapting self-stabilizing census algorithms?)

Perspectives

- Leader election possible in $\mathcal{A} \cap \mathcal{K}_k$ $\mathcal{A}$, but impossible in $\mathcal{A}$: where is the boundary ?
- Find a best trade-off leader election algorithm for $\mathcal{A} \cap \mathcal{K}_k$
- In $\mathcal{A}$, the knowledge of k and n is computationaly equivalent. Is-it still true in bidirectional rings? What about time complexity?
- Self-stabilizing leader election in $\mathcal{U}^* \cap \mathcal{K}_k$ and $\mathcal{A} \cap \mathcal{K}_k$. (research line: adapting self-stabilizing census algorithms?)
- Other topologies: regular graphs, grids, torii, arbitrary connected ...

Perspectives

- Leader election possible in $\mathcal{A} \cap \mathcal{K}_k$ $\mathcal{A}$, but impossible in $\mathcal{A}$: where is the boundary ?
- Find a best trade-off leader election algorithm for $\mathcal{A} \cap \mathcal{K}_k$
- In $\mathcal{A}$, the knowledge of k and n is computationaly equivalent. Is-it still true in bidirectional rings? What about time complexity?
- Self-stabilizing leader election in $\mathcal{U}^* \cap \mathcal{K}_k$ and $\mathcal{A} \cap \mathcal{K}_k$. (research line: adapting self-stabilizing census algorithms?)
- Other topologies: regular graphs, grids, torii, arbitrary connected ...
- Other problems (solutions exist for the consensus problem with permanent failures)

Thank you for your attention

- K. Altisen, A. K. Datta, S. Devismes, A. Durand, and L. L. Larmore. Leader Election in Rings with Bounded Multiplicity (Short Paper). *SSS 2016*, pp. 1-6, Lyon, France, Nov. 7-10, 2016.
- K. Altisen, A. K. Datta, S. Devismes, A. Durand, and L. L. Larmore. Leader Election in Asymmetric Labeled Unidirectional Rings. *IPDPS 2017*, pp. 182-191, Orlando, Florida, USA, May 29 - June 2, 2017.