Election in Unidirectional Rings with Homonyms

Anaïs Durand

February 14, 2023

Université Clermont Auvergne

Joint Work with

Karine Altisen

Stéphane Devismes

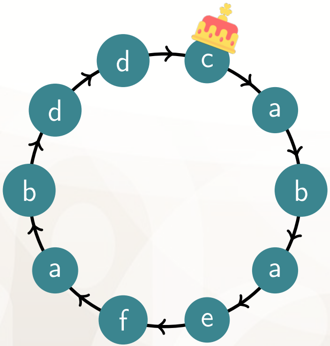
Ajoy K. Datta

Lawrence L. Larmore

- Leader Election in Rings with Bounded Multiplicity (Short paper). *SSS'2016*
- Leader Election in Asymmetric Labeled Unidirectional Rings. *IPDPS'2017*
- Election in Unidirectional Rings with Homonyms. *Journal of Parallel and Distributed Computing*, 2020

UNIVERSITÉ Clermont Auvergne

Context



- Leader election
- Ring networks
- Homonym processes
- Asynchronous message-passing
- Reliable FIFO channels

State of the Art: LE in Rings with Homonyms

- [Flocchini *et. al.*, 04]
  Asynchronous LE in **bidirectional** rings
  with 2 labels, asymmetric labeling and $\mathfrak{n}$ is prime and known

- [Dobrev, Pelc, 04]: Decision on computability + LE
  - **Synchronous** LE in (bidirectional or unidirectional) rings
  - **Asynchronous** LE in **bidirectional** rings

  with knowledge of bounds $\mathfrak{m} \leq \mathfrak{n}$ and $\mathcal{M} \geq \mathfrak{n}$

- [Delporte *et. al.*, 14]:
  Asynchronous LE in **bidirectional** rings where
  number of labels $>$ greatest proper divisor of $\mathfrak{n}$
  - with knowledge of $\mathfrak{n}$
  - without additional knowledge (but only message-terminating)

A different approach: Bounding the number of homonyms

- *Inspired from [Dereniowski, Pelc, 16]:*
  Decision on computability + LE
  in networks of arbitrary topology
  with knowledge of a bound **k** on the multiplicity of a label ℓ.

A different approach:
Bounding the number of homonyms

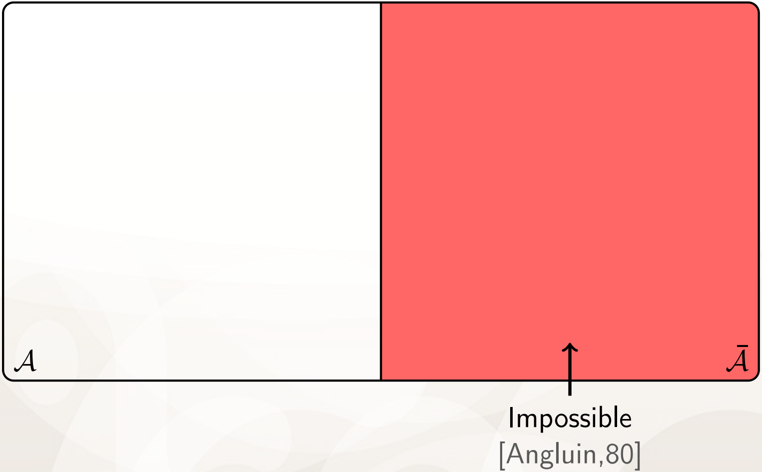
- *Inspired from [Dereniowski, Pelc, 16]:*
  Decision on computability + LE
  in networks of arbitrary topology
  with knowledge of a bound $\mathfrak{k}$ on the multiplicity of a label ℓ.

- *Unidirectional ring classes:*
  - $\mathcal{H}_{\mathfrak{k}}$: multiplicity of a label $\leq \mathfrak{k}$
  - $\mathcal{U}^*$: at least one label is unique
  - $\mathcal{A}$: asymmetric labeling

- *Goal:* Asynchronous process-terminating leader election

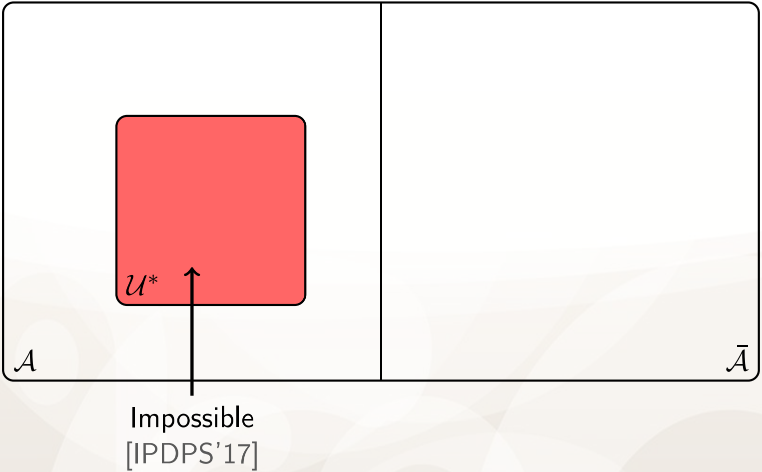
UNIVERSITÉ Clermont Auvergne

Results



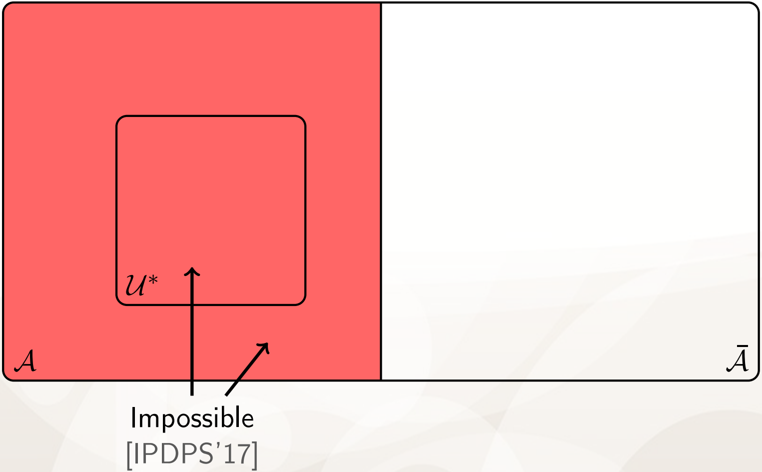
- $\mathcal{H}_{\mathfrak{k}}$: multiplicity of a label $\leq \mathfrak{k}$
- $\mathcal{U}^*$: at least one label is unique
- $\mathcal{A}$: asymmetric labeling

Results



- $\mathcal{H}_{\mathfrak{k}}$: multiplicity of a label $\leq \mathfrak{k}$
- $\mathcal{U}^*$: at least one label is unique
- $\mathcal{A}$: asymmetric labeling

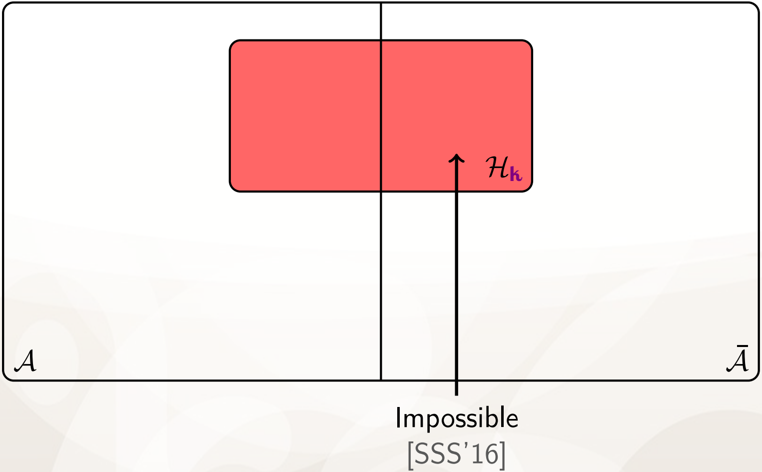
Results



- $\mathcal{H}_{\mathfrak{k}}$: multiplicity of a label $\leq \mathfrak{k}$
- $\mathcal{U}^*$: at least one label is unique
- $\mathcal{A}$: asymmetric labeling

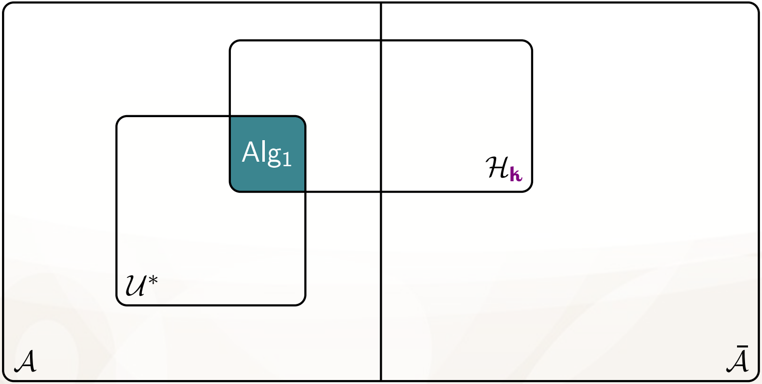
UNIVERSITÉ Clermont Auvergne

Results



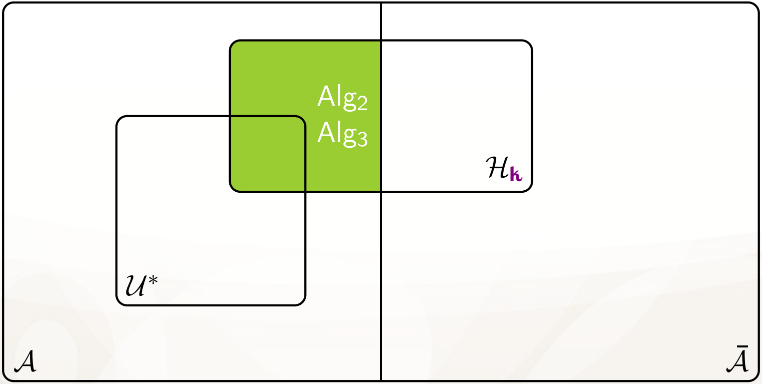
- $\mathcal{H}_{\mathfrak{k}}$: multiplicity of a label $\leq \mathfrak{k}$
- $\mathcal{U}^*$: at least one label is unique
- $\mathcal{A}$: asymmetric labeling

Results



- $\mathcal{H}_{\mathfrak{k}}$: multiplicity of a label $\leq \mathfrak{k}$
- $\mathcal{U}^*$: at least one label is unique
- $\mathcal{A}$: asymmetric labeling

Results

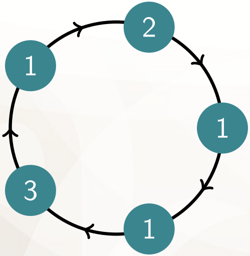


- $\mathcal{H}_{\mathfrak{k}}$: multiplicity of a label $\leq \mathfrak{k}$
- $\mathcal{U}^*$: at least one label is unique
- $\mathcal{A}$: asymmetric labeling

Alg_1 for $\mathcal{U}^* \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the lowest unique label

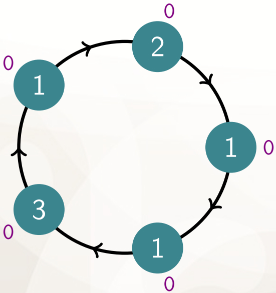
$\mathfrak{k} \geq 3$



Alg_1 for $\mathcal{U}^* \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the lowest unique label

$\mathfrak{k} \geq 3$

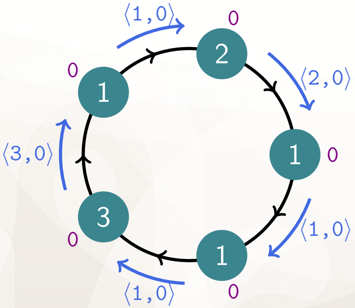


- Counter = rough estimation of the multiplicity

Alg$_1$ for $\mathcal{U}^* \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the lowest unique label

$\mathfrak{k} \geq 3$

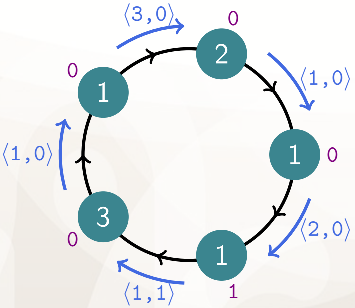


- Counter = rough estimation of the multiplicity

Alg$_1$ for $\mathcal{U}^* \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the lowest unique label

$\mathfrak{k} \geq 3$

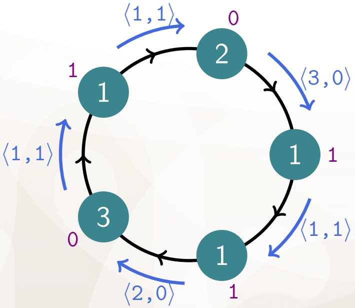


- Counter = rough estimation of the multiplicity

Alg_1 for $\mathcal{U}^* \cap \mathcal{H}_{\mathbf{k}}$

Goal: Electing the lowest unique label

$\mathbf{k} \geq 3$

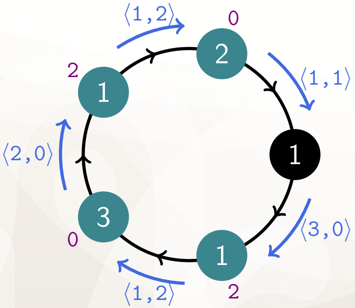


- Counter = rough estimation of the multiplicity
- ***Process elimination:***
  - Lower counter, $\neq$ label $\rightarrow$ label not unique

Alg$_1$ for $\mathcal{U}^* \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the lowest unique label

$\mathfrak{k} \geq 3$

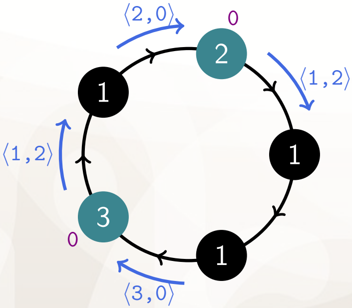


- Counter = rough estimation of the multiplicity
- ***Process elimination:***
  - ▷ Lower counter, $\neq$ label
    $\longrightarrow$ label not unique
- ***Message elimination:***
  - ▷ Passive, same label
    $\longrightarrow$ not relevant

Alg$_1$ for $\mathcal{U}^* \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the lowest unique label

$\mathfrak{k} \geq 3$



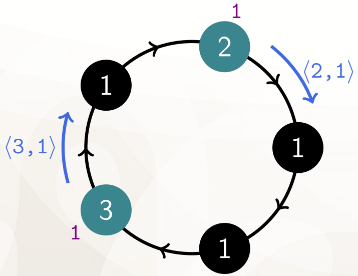
- Counter = rough estimation of the multiplicity
- ***Process elimination:***
  - ▷ Lower counter, $\neq$ label $\longrightarrow$ label not unique
- ***Message elimination:***
  - ▷ Passive, same label $\longrightarrow$ not relevant

UNIVERSITÉ Clermont Auvergne

Alg$_1$ for $\mathcal{U}^* \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the lowest unique label

$\mathfrak{k} \geq 3$

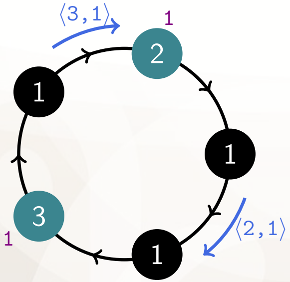


- Counter = rough estimation of the multiplicity
- ***Process elimination:***
  - Lower counter, $\neq$ label $\longrightarrow$ label not unique
- ***Message elimination:***
  - Passive, same label $\longrightarrow$ not relevant

Alg_1 for $\mathcal{U}^* \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the lowest unique label

$\mathfrak{k} \geq 3$

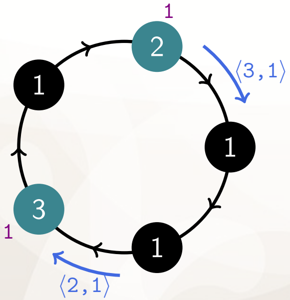


- Counter = rough estimation of the multiplicity
- ***Process elimination:***
  - Lower counter, $\neq$ label $\longrightarrow$ label not unique
- ***Message elimination:***
  - Passive, same label $\longrightarrow$ not relevant

Alg$_1$ for $\mathcal{U}^* \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the lowest unique label

$\mathfrak{k} \geq 3$

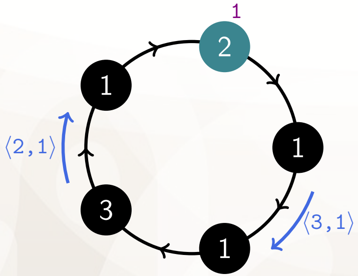


- Counter = rough estimation of the multiplicity
- ***Process elimination:***
  - Lower counter, $\neq$ label
    $\longrightarrow$ label not unique
  - Same counter $\neq 0$, lower label
    $\longrightarrow$ not lowest unique
- ***Message elimination:***
  - Passive, same label
    $\longrightarrow$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

Alg_1 for $\mathcal{U}^* \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the lowest unique label

$\mathfrak{k} \geq 3$

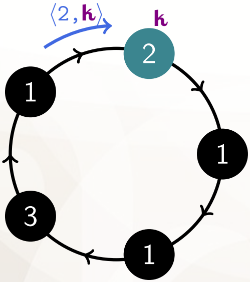


- Counter = rough estimation of the multiplicity
- *Process elimination:*
  - Lower counter, $\neq$ label $\longrightarrow$ label not unique
  - Same counter $\neq 0$, lower label $\longrightarrow$ not lowest unique
- *Message elimination:*
  - Passive, same label $\longrightarrow$ not relevant

Alg_1 for $\mathcal{U}^* \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the lowest unique label

$\mathfrak{k} \geq 3$



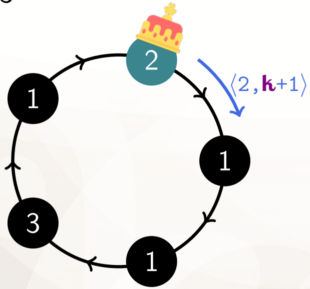
- Counter = rough estimation of the multiplicity
- ***Process elimination:***
  - ▷ Lower counter, $\neq$ label $\longrightarrow$ label not unique
  - ▷ Same counter $\neq 0$, lower label $\longrightarrow$ not lowest unique
- ***Message elimination:***
  - ▷ Passive, same label $\longrightarrow$ not relevant
- ***Election detection:*** receiving $\langle \text{label}, \mathfrak{k} \rangle$

UNIVERSITÉ Clermont Auvergne

Alg_1 for $\mathcal{U}^* \cap \mathcal{H}_{\mathbf{k}}$

Goal: Electing the lowest unique label

$\mathbf{k} \geq 3$



- Counter = rough estimation of the multiplicity
- ***Process elimination:***
  - Lower counter, $\neq$ label
    $\longrightarrow$ label not unique
  - Same counter $\neq 0$, lower label
    $\longrightarrow$ not lowest unique
- ***Message elimination:***
  - Passive, same label
    $\longrightarrow$ not relevant
- ***Election detection:***
  receiving $\langle \text{label}, \mathbf{k} \rangle$

UNIVERSITÉ Clermont Auvergne

Alg$_1$ for $\mathcal{U}^* \cap \mathcal{H}_{\mathtt{k}}$

Asynchronous process-terminating leader election

- *Time complexity:* $O(\mathtt{kn})$ steps

  asymptotically optimal

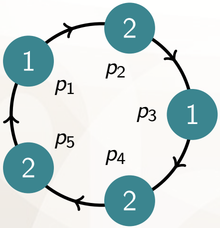
- *Number of messages:* $O(\mathtt{n}^2 + \mathtt{kn})$

- *Memory requirement:* $\lceil \log(\mathtt{k}+1) \rceil + \mathtt{b} + 4$ bits

  where $\mathtt{b}$ = number of bits to store a label

  asymptotically optimal

Alg_2 for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

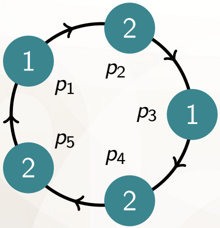


- ***Label sequence at p_1:***

$$\mathfrak{LS}(p_1) = 12212$$

Alg_2 for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the process whose label sequence is a Lyndon Word
Lyndon Word = smallest rotation in lexicographic order



- *Label sequence at p_1:*

$$\mathfrak{LS}(p_1) = 12212$$

Rotations:

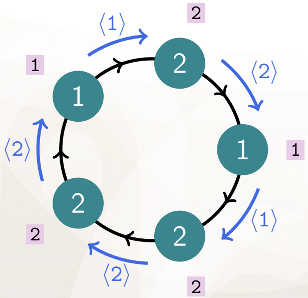
12212 $(= \mathfrak{LS}(p_1))$
21221 $(= \mathfrak{LS}(p_2))$
12122 $(= \mathfrak{LS}(p_3))$ LW $\neq \mathfrak{LS}(p_1)$
21212 $(= \mathfrak{LS}(p_4))$
22121 $(= \mathfrak{LS}(p_5))$

UNIVERSITÉ Clermont Auvergne

Alg$_2$ for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the process whose label sequence is a Lyndon Word
Lyndon Word = smallest rotation in lexicographic order

$\mathfrak{k} \geq 3$

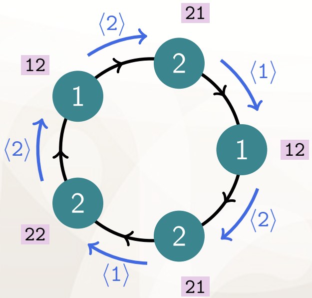


- *Local label aggregation*

Alg$_2$ for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the process whose label sequence is a Lyndon Word
Lyndon Word = smallest rotation in lexicographic order

$\mathfrak{k} \geq 3$

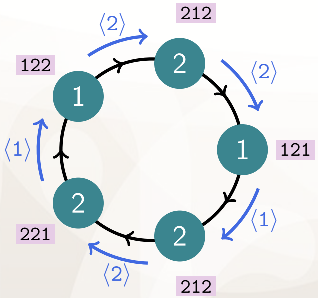


- *Local label aggregation*

Alg_2 for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the process whose label sequence is a Lyndon Word
Lyndon Word = smallest rotation in lexicographic order

$\mathfrak{k} \geq 3$

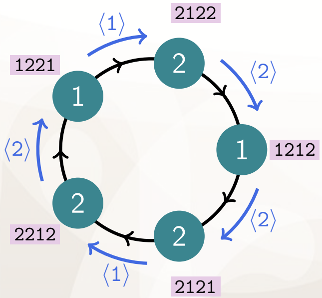


- ***Local label aggregation***

Alg_2 for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the process whose label sequence is a Lyndon Word
Lyndon Word = smallest rotation in lexicographic order

$\mathfrak{k} \geq 3$

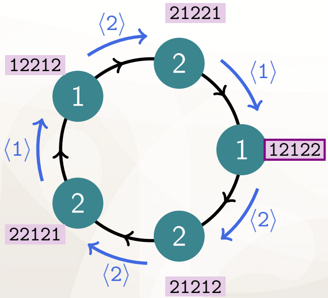


- ***Local label aggregation***

Alg_2 for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the process whose label sequence is a Lyndon Word
Lyndon Word = smallest rotation in lexicographic order

$\mathfrak{k} \geq 3$

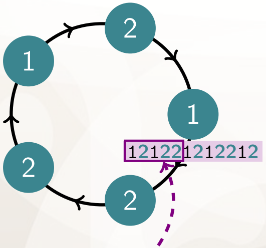


- *Local label aggregation*
- $\mathfrak{n}$ is not known
  $\longrightarrow$ no detection of election yet

Alg$_2$ for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the process whose label sequence is a Lyndon Word
Lyndon Word = smallest rotation in lexicographic order

$\mathfrak{k} \geq 3$



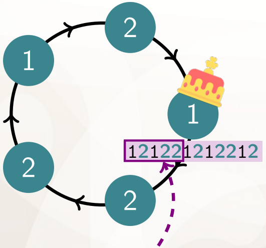
Smallest repeating prefix = $\mathfrak{LS}$

- *Local label aggregation*
- $\mathfrak{n}$ is not known
  $\rightarrow$ no detection of election yet
- *Termination detection:*
  $(2\mathfrak{k}+1)\times$ the same label
  $\rightarrow \geq 2\times$ the label sequence

Alg$_2$ for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Electing the process whose label sequence is a Lyndon Word
Lyndon Word = smallest rotation in lexicographic order

$\mathfrak{k} \geq 3$



Smallest repeating prefix = $\mathfrak{LS}$
= Lyndon Word

- *Local label aggregation*
- $\mathfrak{n}$ is not known
  $\rightarrow$ no detection of election yet
- *Termination detection:*
  $(2\mathfrak{k}+1)\times$ the same label
  $\rightarrow \geq 2\times$ the label sequence

Alg_2 for $\mathcal{A} \cap \mathcal{H}_{\mathbf{k}}$

Asynchronous process-terminating leader election

- *Time complexity:* $O(\mathbf{kn})$ steps

  asymptotically optimal

- *Number of messages:* $O(\mathbf{kn}^2)$

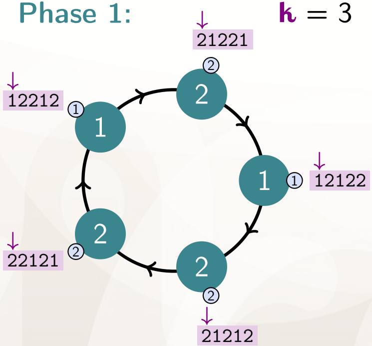
- *Memory requirement:* $O(\mathbf{knb})$ bits

  where $\mathbf{b}$ = number of bits to store a label

Alg_3 for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Reducing the memory requirement of Alg_2 using Peterson principle with radix sort

Phase 1: $\mathfrak{k} = 3$

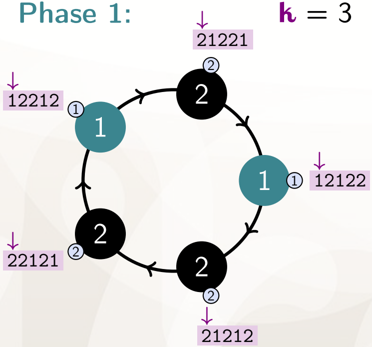


- *During a phase:* not lowest value of active processes $\rightarrow$ process eliminated

Alg_3 for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Reducing the memory requirement of Alg_2 using Peterson principle with radix sort

Phase 1: $\mathfrak{k} = 3$



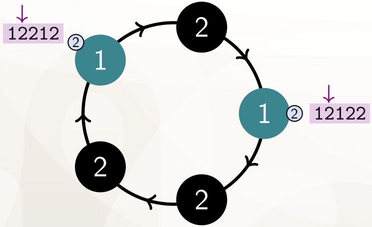
- ***During a phase:***
  not lowest value of active processes
  $\rightarrow$ process eliminated

UNIVERSITÉ Clermont Auvergne

Alg_3 for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Reducing the memory requirement of Alg_2 using Peterson principle with radix sort

Phase 2: $\mathfrak{k} = 3$

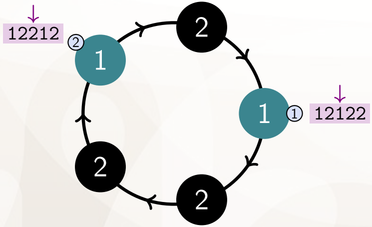


- *During a phase:*
  not lowest value of active processes
  $\rightarrow$ process eliminated

Alg_3 for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Reducing the memory requirement of Alg_2 using Peterson principle with radix sort

Phase 3: $\mathfrak{k} = 3$

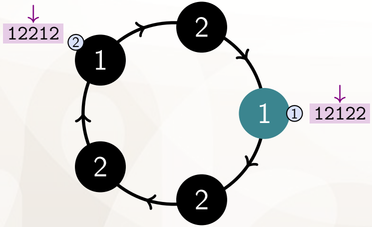


- *During a phase:*
  not lowest value of active processes
  $\rightarrow$ process eliminated

Alg$_3$ for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Reducing the memory requirement of Alg$_2$ using Peterson principle with radix sort

Phase 3: $\mathfrak{k} = 3$



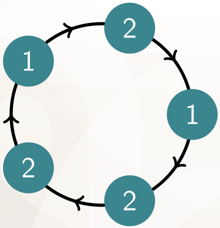
- *During a phase:*
  not lowest value of active processes
  $\longrightarrow$ process eliminated

Université Clermont Auvergne

Alg_3 for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Reducing the memory requirement of Alg_2 using Peterson principle with radix sort

$\mathfrak{k} = 3$



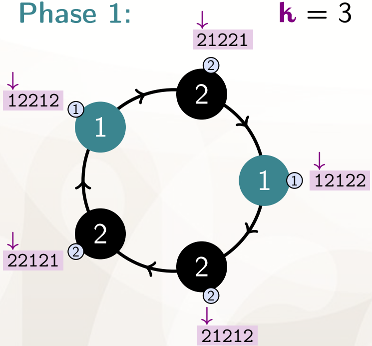
- *During a phase:*
  not lowest value of active processes
  $\longrightarrow$ process eliminated

- *Phase switch:*
  sending its value to its neighbor

Alg_3 for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Reducing the memory requirement of Alg_2 using Peterson principle with radix sort

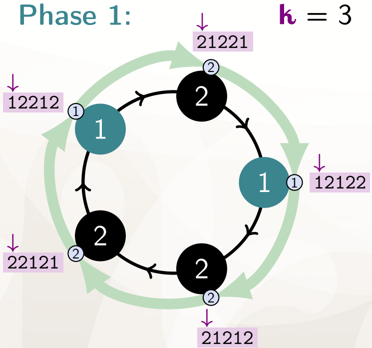
Phase 1: $\mathfrak{k} = 3$



- *During a phase:* not lowest value of active processes $\rightarrow$ process eliminated
- *Phase switch:* sending its value to its neighbor

Alg$_3$ for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Reducing the memory requirement of Alg$_2$ using Peterson principle with radix sort

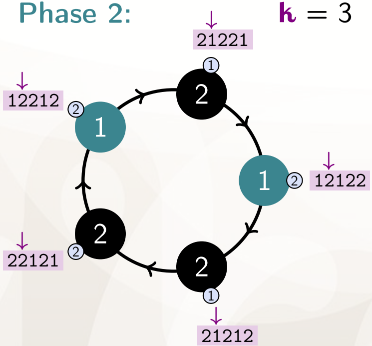


- *During a phase:*
  not lowest value of active processes
  $\longrightarrow$ process eliminated
- *Phase switch:*
  sending its value to its neighbor

Alg_3 for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Reducing the memory requirement of Alg_2 using Peterson principle with radix sort

Phase 2: $\mathfrak{k} = 3$



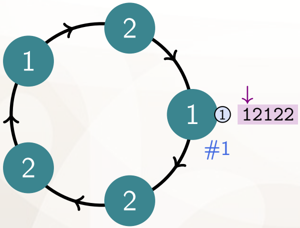
- *During a phase:*
  not lowest value of active processes
  $\longrightarrow$ process eliminated
- *Phase switch:*
  sending its value to its neighbor
  when? $\mathfrak{k}+1$ times current value

Université Clermont Auvergne

Alg_3 for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Reducing the memory requirement of Alg_2 using Peterson principle with radix sort

Phase 1: $\mathfrak{k} = 3$

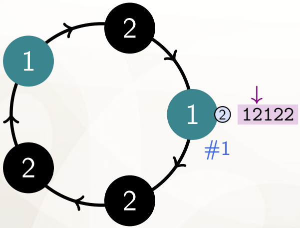


- *During a phase:*
  not lowest value of active processes
  $\rightarrow$ process eliminated
- *Phase switch:*
  sending its value to its neighbor
  when? $\mathfrak{k} + 1$ times current value
- *Termination detection:*
  $\mathfrak{k} + 1$ times its label as value

Alg$_3$ for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Reducing the memory requirement of Alg$_2$ using Peterson principle with radix sort

Phase 2: $\mathfrak{k} = 3$

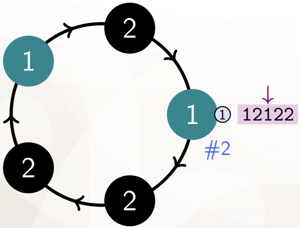


- *During a phase:*
  not lowest value of active processes
  $\rightarrow$ process eliminated
- *Phase switch:*
  sending its value to its neighbor
  when? $\mathfrak{k}+1$ times current value
- *Termination detection:*
  $\mathfrak{k}+1$ times its label as value

Alg$_3$ for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Reducing the memory requirement of Alg$_2$ using Peterson principle with radix sort

Phase 3: $\mathfrak{k} = 3$

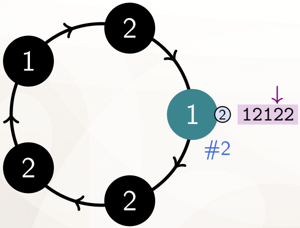


- *During a phase:*
  not lowest value of active processes
  $\rightarrow$ process eliminated
- *Phase switch:*
  sending its value to its neighbor
  when? $\mathfrak{k}+1$ times current value
- *Termination detection:*
  $\mathfrak{k}+1$ times its label as value

Alg_3 for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Reducing the memory requirement of Alg_2 using Peterson principle with radix sort

Phase 4: $\mathfrak{k} = 3$

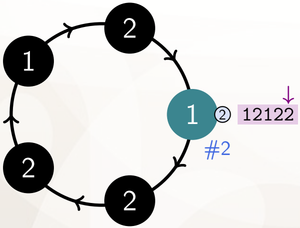


- *During a phase:*
  not lowest value of active processes
  $\rightarrow$ process eliminated
- *Phase switch:*
  sending its value to its neighbor
  when? $\mathfrak{k}+1$ times current value
- *Termination detection:*
  $\mathfrak{k}+1$ times its label as value

Alg$_3$ for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Reducing the memory requirement of Alg$_2$ using Peterson principle with radix sort

Phase 5: $\mathfrak{k} = 3$

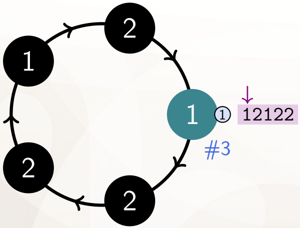


- *During a phase:*
  not lowest value of active processes $\longrightarrow$ process eliminated
- *Phase switch:*
  sending its value to its neighbor
  when? $\mathfrak{k}+1$ times current value
- *Termination detection:*
  $\mathfrak{k}+1$ times its label as value

Alg_3 for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Reducing the memory requirement of Alg_2 using Peterson principle with radix sort

Phase 6: $\mathfrak{k} = 3$

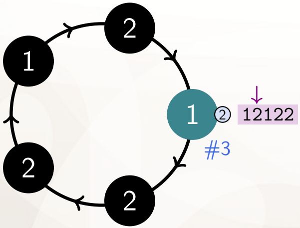


- *During a phase:*
  not lowest value of active processes
  $\rightarrow$ process eliminated
- *Phase switch:*
  sending its value to its neighbor
  when? $\mathfrak{k}+1$ times current value
- *Termination detection:*
  $\mathfrak{k}+1$ times its label as value

Alg_3 for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Reducing the memory requirement of Alg_2 using Peterson principle with radix sort

Phase 7: $\mathfrak{k} = 3$

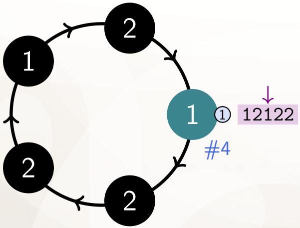


- *During a phase:*
  not lowest value of active processes
  $\rightarrow$ process eliminated
- *Phase switch:*
  sending its value to its neighbor
  when? $\mathfrak{k} + 1$ times current value
- *Termination detection:*
  $\mathfrak{k} + 1$ times its label as value

Alg_3 for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Reducing the memory requirement of Alg_2 using Peterson principle with radix sort

Phase 8: $\mathfrak{k} = 3$

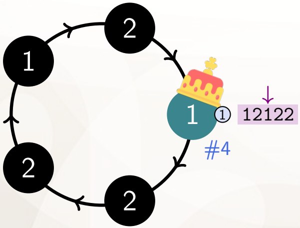


- *During a phase:*
  not lowest value of active processes
  $\rightarrow$ process eliminated
- *Phase switch:*
  sending its value to its neighbor
  when? $\mathfrak{k} + 1$ times current value
- *Termination detection:*
  $\mathfrak{k} + 1$ times its label as value

Alg_3 for $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

Goal: Reducing the memory requirement of Alg_2 using Peterson principle with radix sort

Phase 8: $\mathfrak{k} = 3$



- *During a phase:*
  not lowest value of active processes
  $\rightarrow$ process eliminated
- *Phase switch:*
  sending its value to its neighbor
  when? $\mathfrak{k} + 1$ times current value
- *Termination detection:*
  $\mathfrak{k} + 1$ times its label as value

Alg_3 for $\mathcal{A} \cap \mathcal{H}_{\mathtt{k}}$

Asynchronous process-terminating leader election

- *Time complexity:* $O(\mathtt{k}^2\mathtt{n}^2)$ steps
- *Number of messages:* $O(\mathtt{k}^2\mathtt{n}^2)$
- *Memory requirement:* $2\lceil\log \mathtt{k}\rceil + 3\mathtt{b} + 5$ bits
  where $\mathtt{b}$ = number of bits to store a label
  asymptotically optimal

Université Clermont Auvergne

Conclusion

Process-terminating leader election for unidirectional rings with label multiplicity bounded by $\mathfrak{k}$ and

- at least one unique label $\mathcal{U}^* \cap \mathcal{H}_{\mathfrak{k}}$

	Time	*Memory*
Alg.1	asymptotically optimal	asymptotically optimal

- asymmetric labeling $\mathcal{A} \cap \mathcal{H}_{\mathfrak{k}}$

	Time	*Memory*
Alg.2	asymptotically optimal	large
Alg.3	large	asymptotically optimal

UNIVERSITÉ Clermont Auvergne