OPODIS 2018

Self-Stabilizing Token Distribution with Constant-Space for Trees

Yuichi Sudo[1], Ajoy K. Datta[2],
Lawrence L. Larmore[2], Toshimitsu Masuzawa[1]

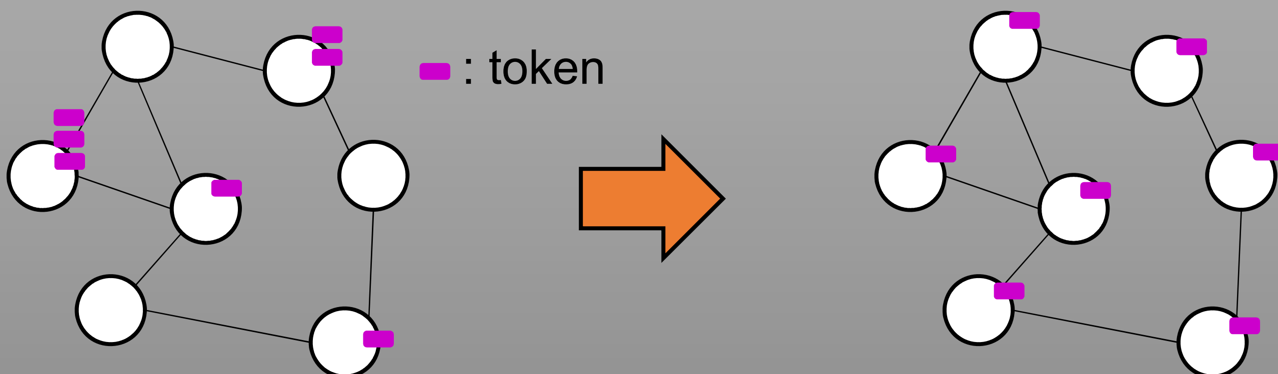
1. Osaka University, Japan

2. The University of Nevada, Las Vegas, USA

Token Distribution Problem

- Originally defined by Peleg and Upfal in1989 [4]

- Initially: $\boldsymbol{n}$ **tokens** are arbitrarily distributed (n: #nodes)
- GOAL: Eech node has **exactly one token**



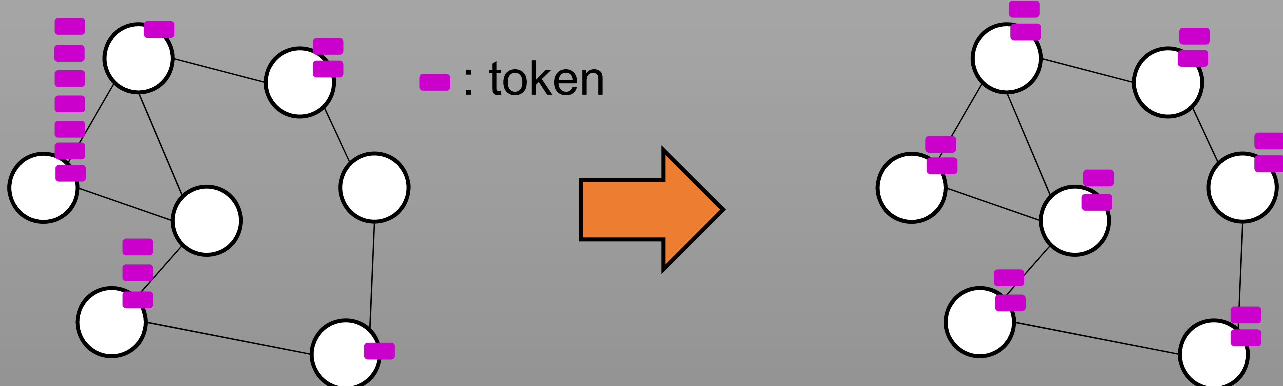
- Constraints: A node holds **at most $\boldsymbol{l}$** tokens at any time
  (l: token space capacity at a node)

[4] Peleg, D., Upfal, E.: The token distribution problem. SIAM J. Comput. 18(2), 229–243 (1989)

Generalized Token Distribution

- Initially: $\boldsymbol{nk}$ **tokens** are arbitrarily distributed (n: #nodes)
- GOAL: Each node has **exactly $\boldsymbol{k}$ tokens**

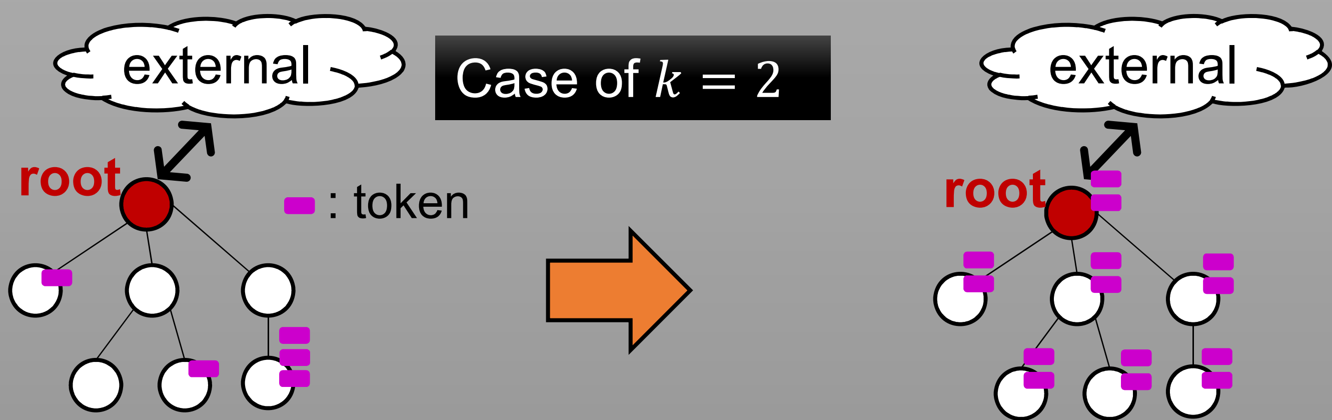
Case of $k = 2$



- Constraints: A node holds **at most $\boldsymbol{l}$** tokens at any time (l: token space capacity at a node)

Self-stabilizing (SS) Token Distribution

- Initially: **Any number of tokens** are **arbitrarily** distributed.
  Each process has an arbitrary state.
- GOAL: Each node has **exactly k tokens** (n: #nodes)



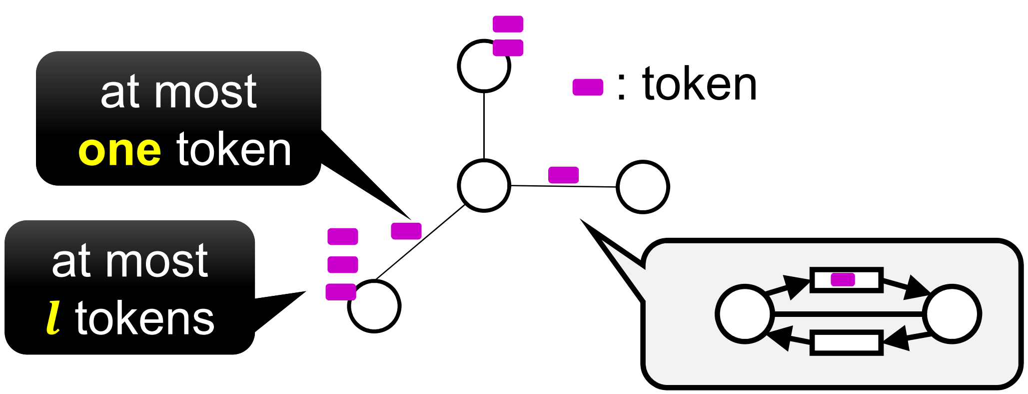
- Constraints: A node holds **at most l tokens** at any time
  (l: token space capacity at a node)

Assumption

- **Rooted tree** networks
- The root can push/pull tokens to/from **the external store**
- Each node knows k

Model

- Asynchronous trees
- Link register model
  - Two registers at each link, one for each direction
  - nodes can communicate only through the registers
- Token space capacity
  - A node has a token space for **at most l tokens**
  - A link register has a token space for **at most one token**
    - Tokens are **transferred one by one**



Efficiency metrics

- Convergence time
  - Evaluated in asynchronous rounds
- Number of **redundant** token moves
  - #(token moves) – (the optimum number of token moves)
  - i.e., the number of **unnecessary** token moves
- Work space of a node and a register
  - Space for all variables **other than tokens**

Our Contributions: SS token distribution

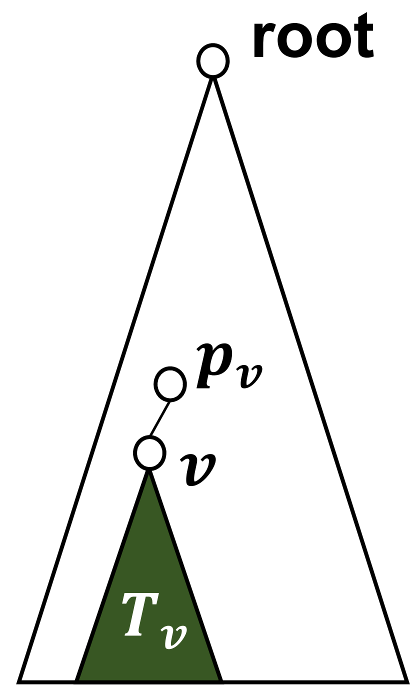
	convergence time	#(redundant token moves)	work space	
			node	link
Base	$\boldsymbol{O(nl)}$	$O(nh\varepsilon)$	**0**	$\boldsymbol{O(1)}$
SyncDist	$\boldsymbol{O(nl)}$	$O(nh)$	$\boldsymbol{O(1)}$	$\boldsymbol{O(1)}$
PIFDist	$\boldsymbol{O(nhl)}$	$\boldsymbol{O(n)}$	$\boldsymbol{O(1)}$	$\boldsymbol{O(1)}$
Lower bounds	$\Omega(nl)$	$\Omega(n)$	-	-

$\varepsilon = \min(k, l - k)$

- Unfair daemon
- **Constant work space**
- **Base**, **SyncDist**: **optimal convergence time**
- **SyncDist**: improved #(redundant token moves)
- **PIFDist**: **optimal #(redundant token moves)**
  at the expense of increased convergence time

Strategy

- Each node v determine whether or not **its subtree T_v** has **excess**/**shortage** of tokens (i.e., **more than** / **less than** $k \cdot |T_k|$ tokens)



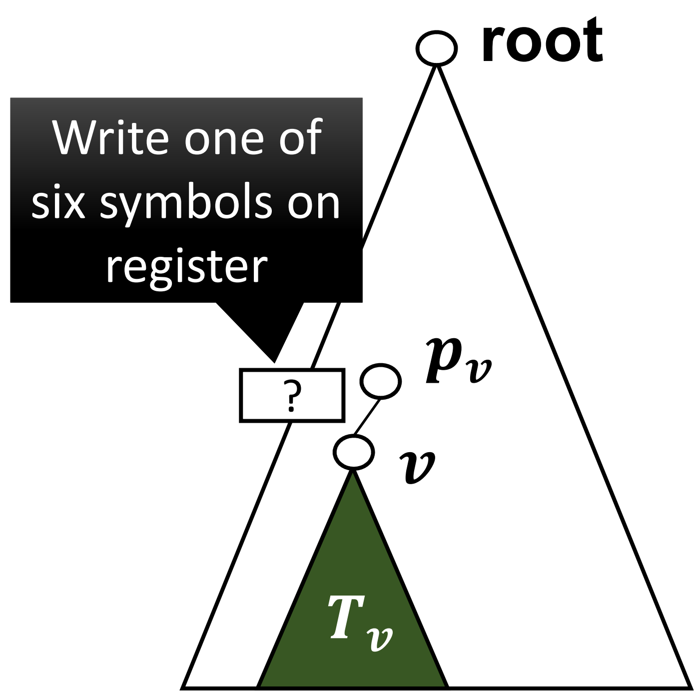
- **Excess → v sends a token to parent p_v**
- **Shortage → v asks p_v to send a token to v**
- **Balanced → Do nothing**
- Root resolves the excess or shortage of the whole tree by pushing to or pulling from the external store

If every node **always** detects the excess/shortage **correctly**,
- redundant token moves never happen
- token distribution is eventually achieved

Strategy for **constant work space**

- Each node tries to detect the excess/shortage of its subtree **without counting #tokens**
- Use **6 symbols** to inform the parent of excess/shortage in T_v

Symbol	Meaning
+1	**excess**
$\mathbf{0^+}$	excess or balanced
0	balanced
$\mathbf{0^-}$	shortage or balanced
-1	**shortage**
$\perp$	unsure

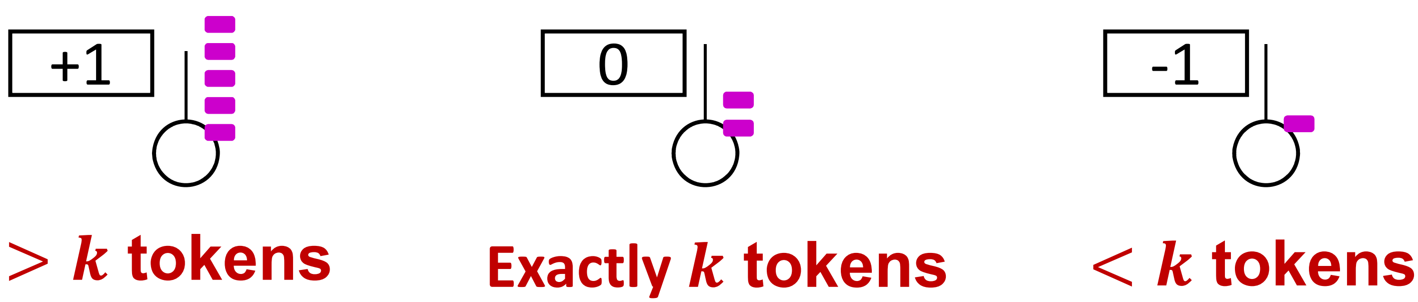


- Node v sends a token to its parent if its symbol is $+1$
- Parent p_v sends a token to v if v's symbol is -1

Determine a Symbol for a **Leaf** Node

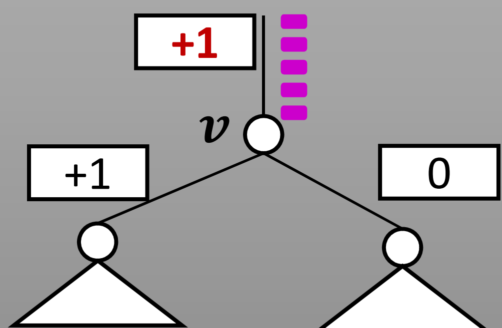
- Each leaf can easily determine its symbol because its subtree consists of only itself

Case of k =2

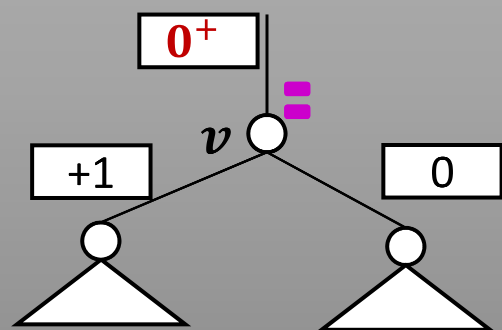


Determine a Symbol for a **Non-Leaf** Node

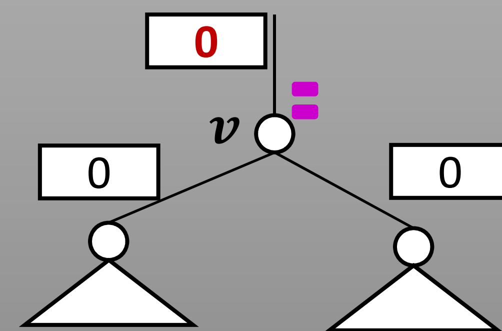
- **$> k$ tokens** in v
- All children outputs $+1, 0^+$, or 0



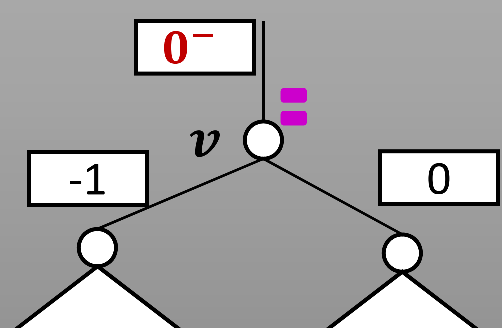
- **Exactly k tokens** in v
- All children outputs $+1, 0^+$, or 0



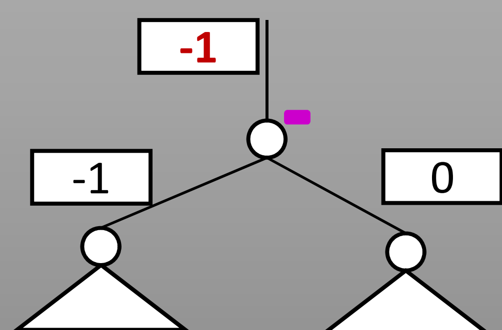
- **Exactly k tokens** in v
- All children outputs 0



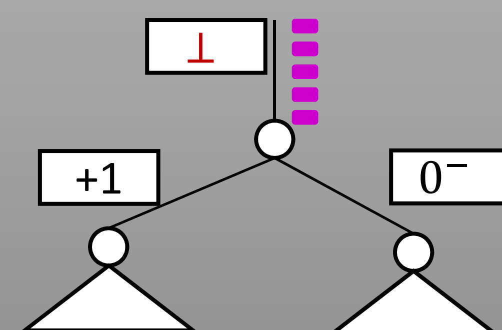
- **Exactly k tokens** in v
- All children outputs 0, 0^-, or -1



- **$< k$ tokens** in v
- All children outputs 0, 0^-, or -1



- Otherwise



Good Properties

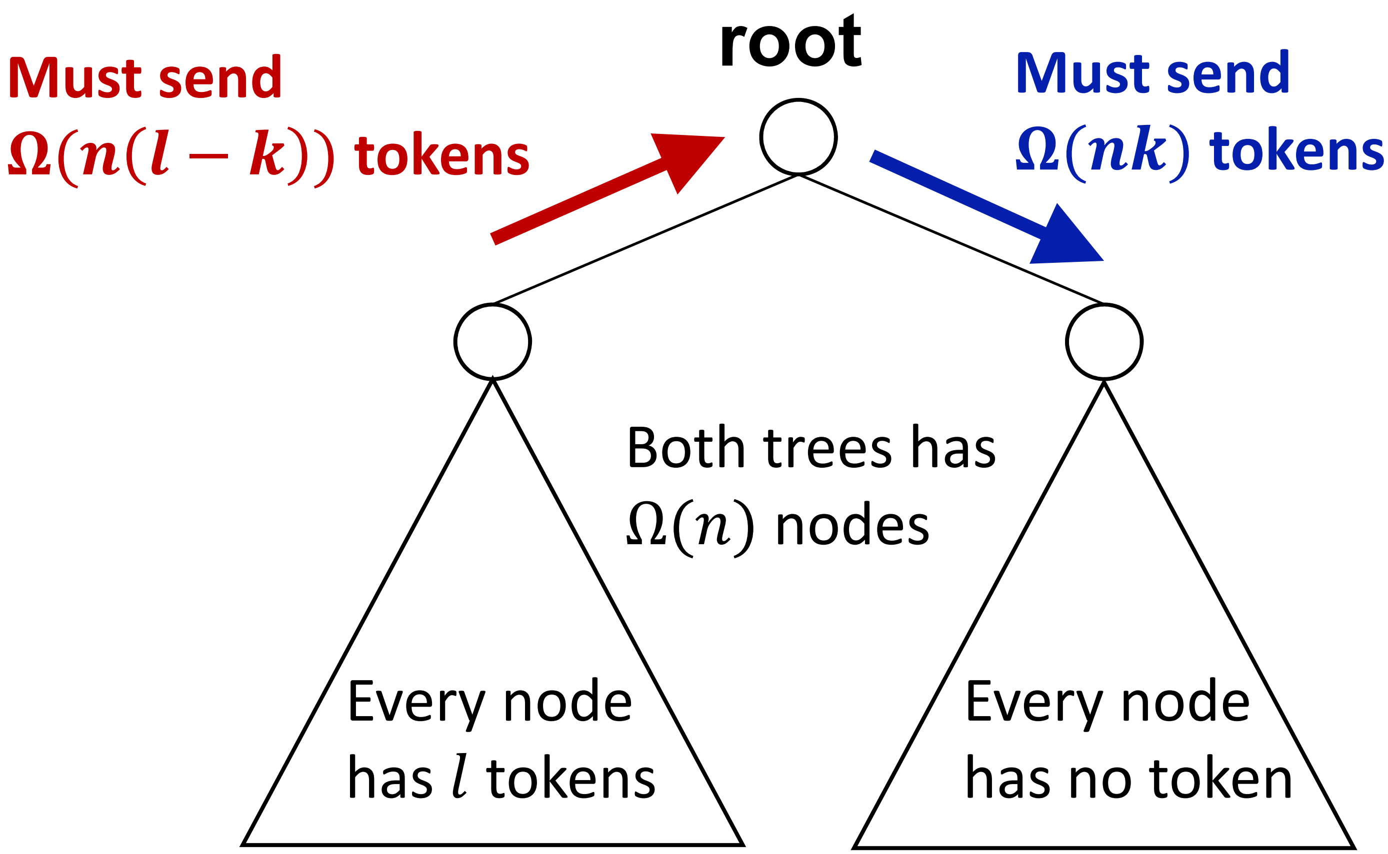
- A node is called ***consistent*** when its symbol does not contradict to the actual excess/shortage in its sub-tree
- A configuration γ is ***consistent*** if every node is consistent in γ

+1	**excess**
0^+	excess or balanced
0	balanced
0^-	shortage or balanced
-1	**shortage**
$\perp$	unsure

h: the height of the tree

- Starting from any configuration, **every execution reaches a consistent configuration after $O(h)$ rounds**
- Thereafter, **redundant token moves never happen** ☺
- Furthermore, at least one token moves in every $O(1)$ rounds
  → Token distribution is achieved within $O(nhl)$ rounds
- Careful analysis proves that **$O(nl)$ rounds are enough**

$\Omega(nl)$ rounds are necessary



$\Omega(\max(n(l-k), nk)) = \Omega(nl)$ rounds
are required to achieve token distribution

Three algorithms

- **Base**
  - Simply implement the above strategies
  - **Optimal conv. time $\boldsymbol{O(nl)}$**, redund. token moves $O(nh\epsilon)$
    h: tree height, $\varepsilon = \min(k, l-k)$
- **SyncDist**
  - Use a **self-stabilizing (loose-)synchronizer** (Datta 15) so that each node can send $O(h)$ tokens in $O(h)$ rounds
  - **Optimal conv. time $\boldsymbol{O(nl)}$**, redund. token moves $O(nh)$
- **PIFDist**
  - Use a **self-stabilizing PIF** (Bui 07) so that each node can send $O(1)$ tokens in $O(h)$ rounds
  - **Conv. time $\boldsymbol{O(nlh)}$**, **optmal redund. token moves $\boldsymbol{O(n)}$**

Conclusions: SS token distribution

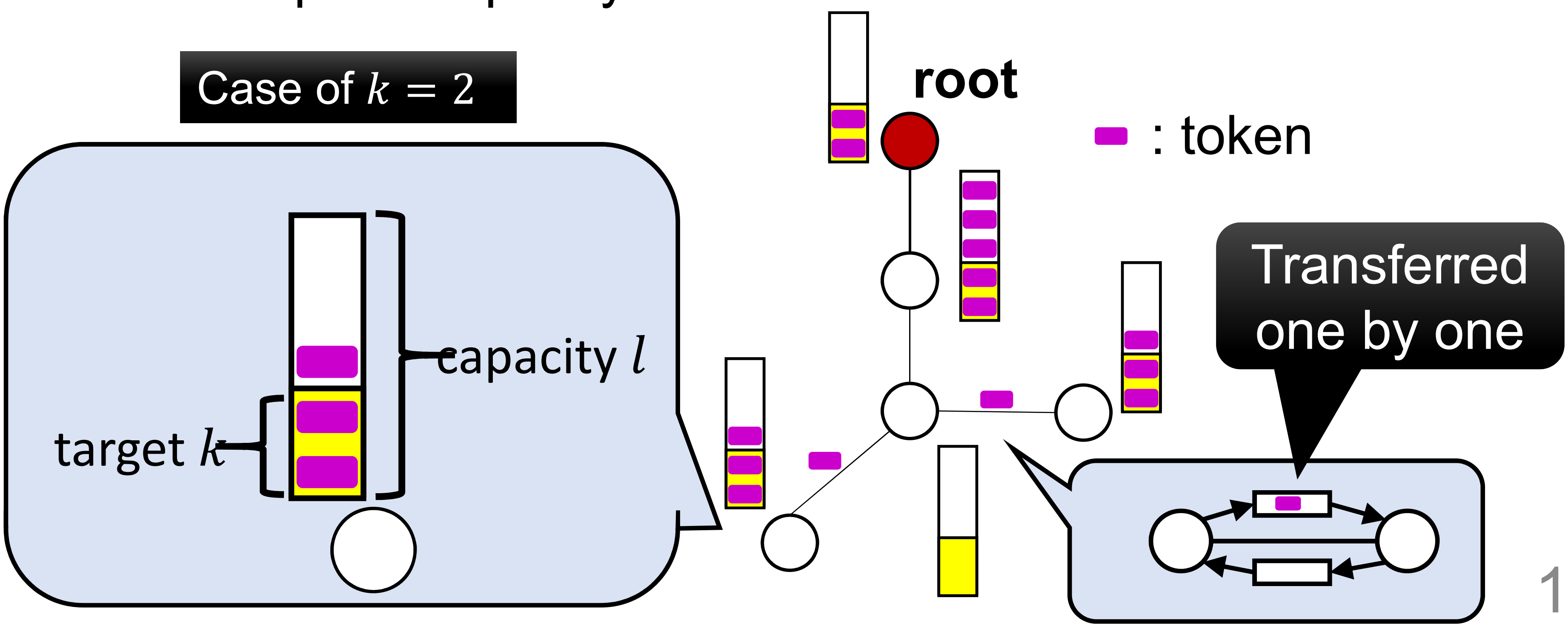
	convergence time	#(redundant token moves)	work space	
			node	link
Base	$\boldsymbol{O(nl)}$	$O(nh\varepsilon)$	**0**	$\boldsymbol{O(1)}$
SyncDist	$\boldsymbol{O(nl)}$	$O(nh)$	$\boldsymbol{O(1)}$	$\boldsymbol{O(1)}$
PIFDist	$\boldsymbol{O(nhl)}$	$\boldsymbol{O(n)}$	$\boldsymbol{O(1)}$	$\boldsymbol{O(1)}$
Lower bounds	$\Omega(nl)$	$\Omega(n)$	-	-

$\varepsilon = \min(k, l - k)$

- Unfair daemon
- **Constant work space**
- **Base**, **SyncDist**: **optimal convergence time**
- **SyncDist**: improved #(redundant token moves)
- **PIFDist**: **optimal #(redundant token moves)**
  at the expense of increased convergence time

Model

- Asynchronous rooted tree
- Link register model
  - Two registers at each link, one for each direction
  - nodes can communicate only through the registers
- Token space capacity



Token moves

- Each node v sends tokens following the two rules
  - Finds a child with symbol -1 → Sends a token to the child
  - v's symbol is $+1$ → Sends a token to v's parent