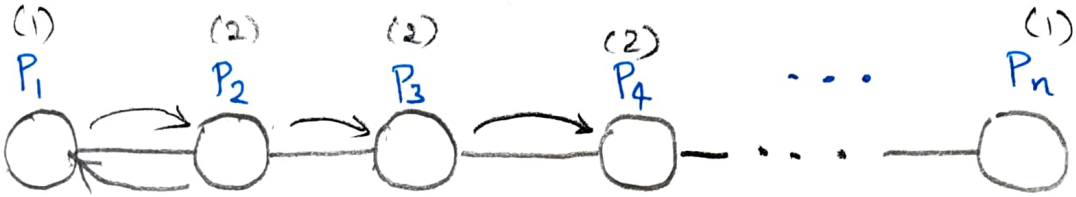
An Alternative time-optimal Algo for distributed Sorting on a line network

24 Jan 2022

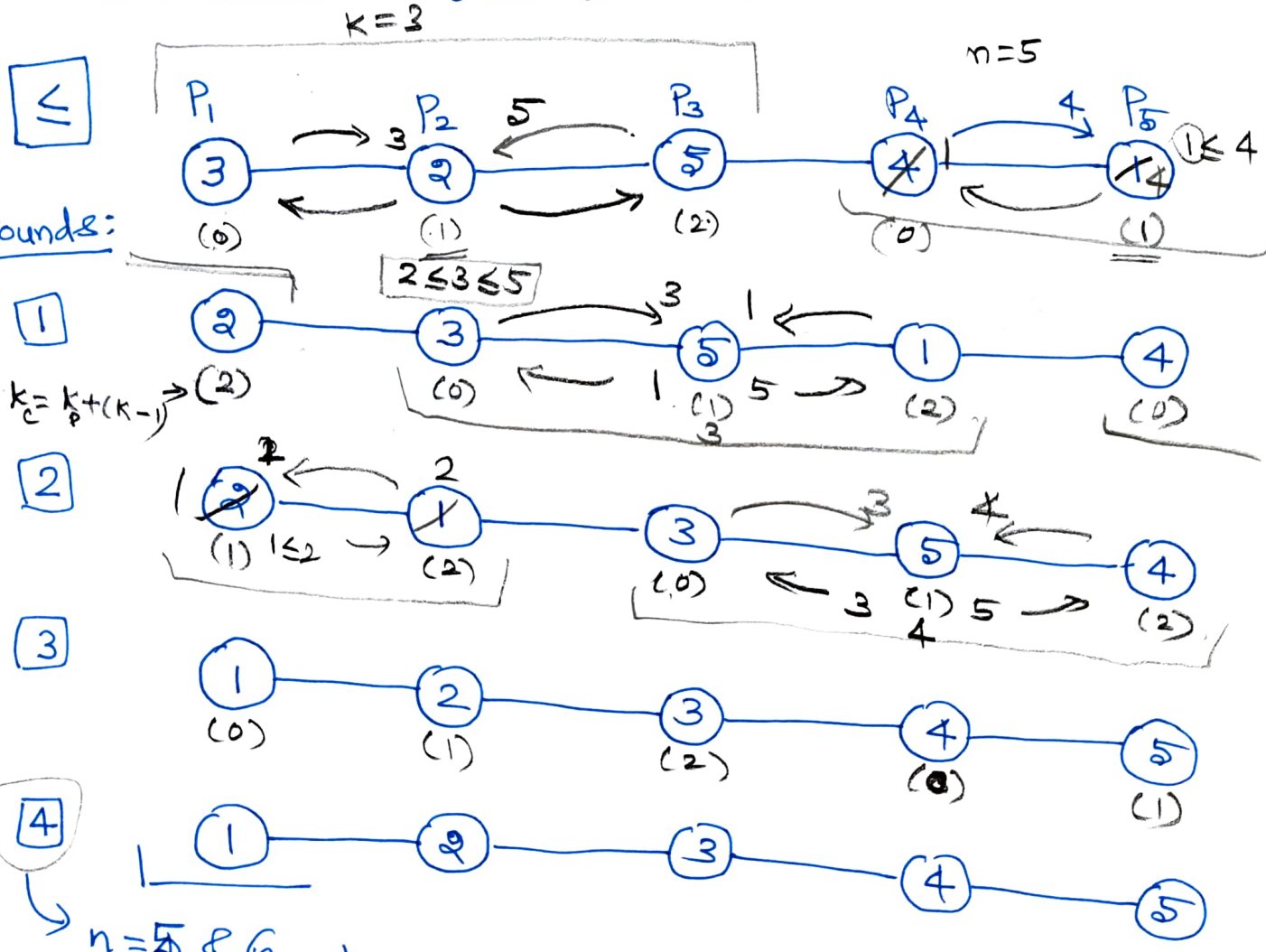
n-rounds - odd-even

$(n-1)$-rounds - Sasaki's.

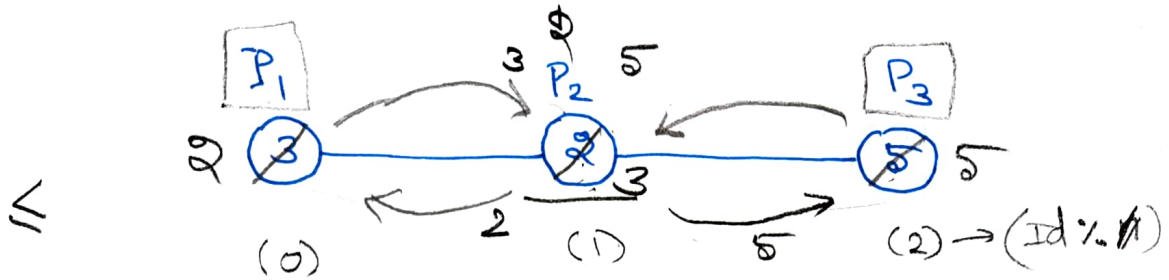


* Comm. channel is reliable
* Processes are fault-free

} Network is fault-free



$n = 5$ & $(n-1) = 4$



≤

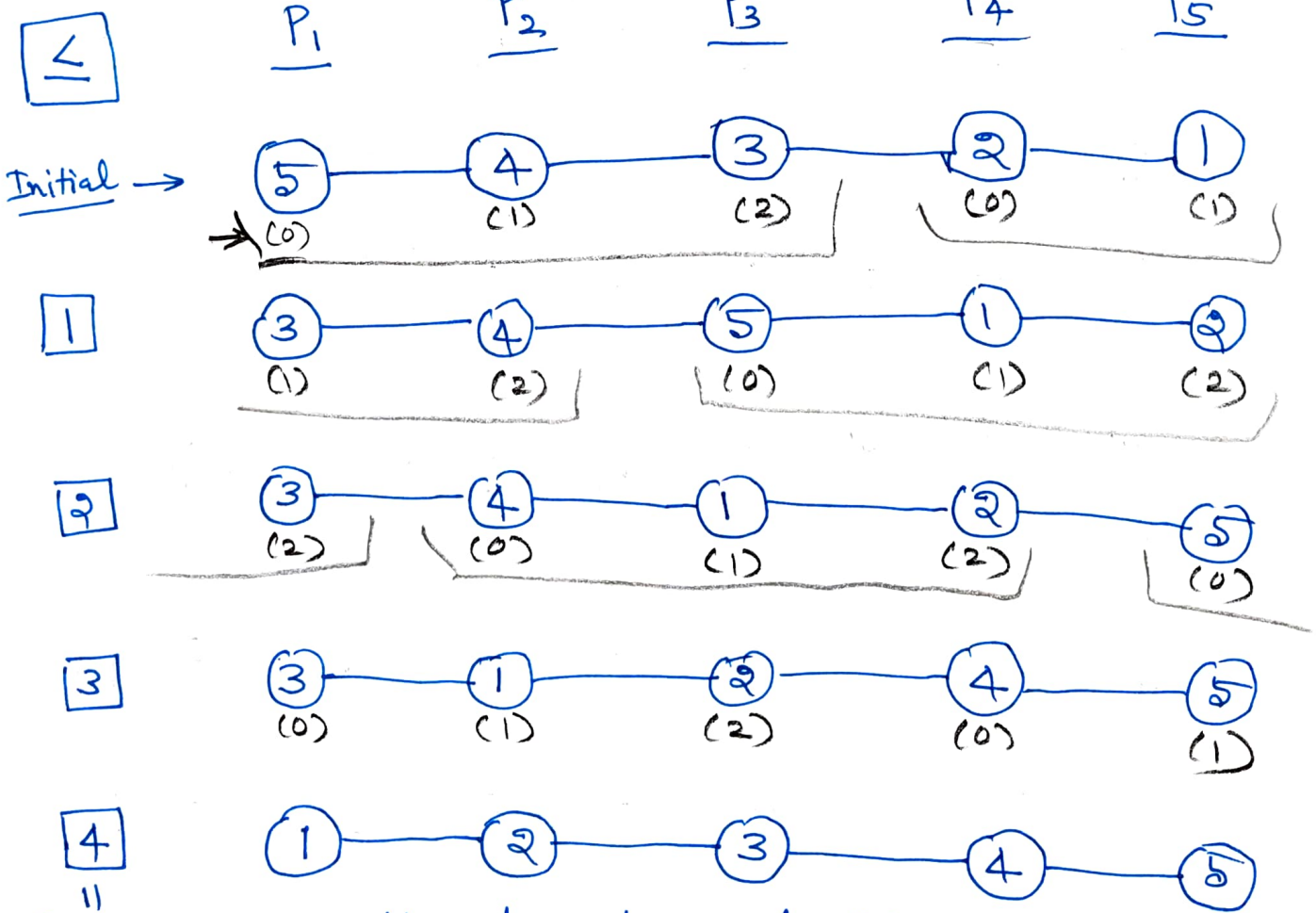
3, 2, 5

≤

2, 3, 5

(2 ≤ 3 ≤ 5)

EX:



(n-1) rounds

→ You do not need Soln. Selection strategy

→ need only (n-1) rounds

→ # msgs per round remains same as of Sasaki's.

→ Faster running time

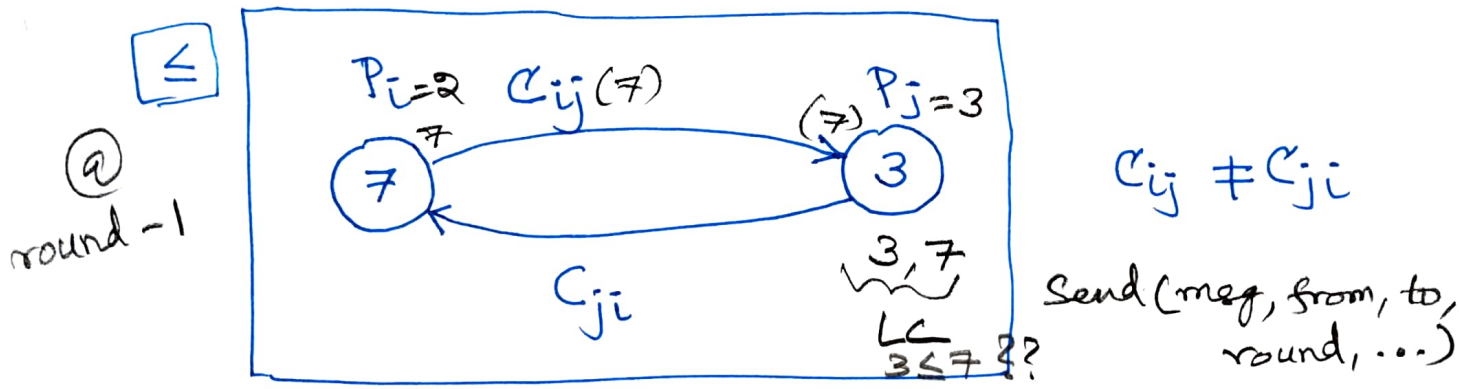
	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
	5	20	7	9	1	17	11	18
$n=8$	(1)	(2)	(0)	(1)	(2)	(0)	(1)	(2)
①	20 (2)	5 (0)	9 (1)	7 (2)	1 (0)	18 (1)	17 (2)	11 (0)
②	20 (0)	9 (1)	7 (2)	5 (0)	18 (1)	17 (2)	1 (0)	11 (1)
③	20 (1)	9 (2)	7 (0)	18 (1)	17 (2)	5 (0)	11 (1)	1 (2)
④	20 (2)	9	18	17	7	11	5	1
⑤	20 (0)	18	17	9	11	7	5	1
⑥	20 (1)	18 (2)	17	11	9	7	5	1
⑦	20	≥ 18	≥ 17	≥ 11	≥ 9	≥ 7	≥ 5	≥ 1

$= (n-1)$ rounds

⊛ Model Discrete Events:

Assignment: 1

a) Implement odd-even transposition Sort
b) Implement Sasaki's sorting Algo
c) Implement ~~the~~ alterative time-optimal (Median based) DSA.



* Internal Events (lc)
* External Events (lc, Send, receive)

→ P_i sends a message (7) to P_j
→ P_j receives a message (7) from P_i

→ local comp:
Compute min, max
3 ≤ 7
min = 3, max = 7
Send 3 to P_i
update 7 @ P_j

→ send(7, 2, 3, 1); ← in which round
← receive(7, 3, 2, 1);
msg, who, from whom
@ P_j
Compute 3 ≤ 7
find min, max
min = 3, max = 7
update val(P_j) = 7 (max)
→ Send(min, 3, 2, 1);
← receive(3, 2, 3, 1);

* Compute @ P_i:
val(P_i) = min
// 3 ← 3