Algorithmic Approach to D.S.

2020/2021

20/09/21

LEADER ELECTION

COMPLETE GRAPH LB

- WORST CASE
  - Process i receive IDs in increasing order
  - it receive $n-1$ msgs
  - it broadcast ($n-1$ msgs) for each one he receive
  - there are n processors

  ⇒ #MSGS = $(n-1)(n-1)n = O(n^3)$
- BEST CASE:
  - Process i receives IDs in non-increasing order
  - it receive $n-1$ msgs
  - there are n processors

  ⇒ #MSGS = $(n-1)\,n = O(n^2)$

CHANG/ROBERTS

- ASSUME that exists a UNIDIRECTIONAL RING of non faulty processors
- PROCLAMATION is needed (<End> msg)

ALGO

Ip. {M = ∅}

M = P
SEND M

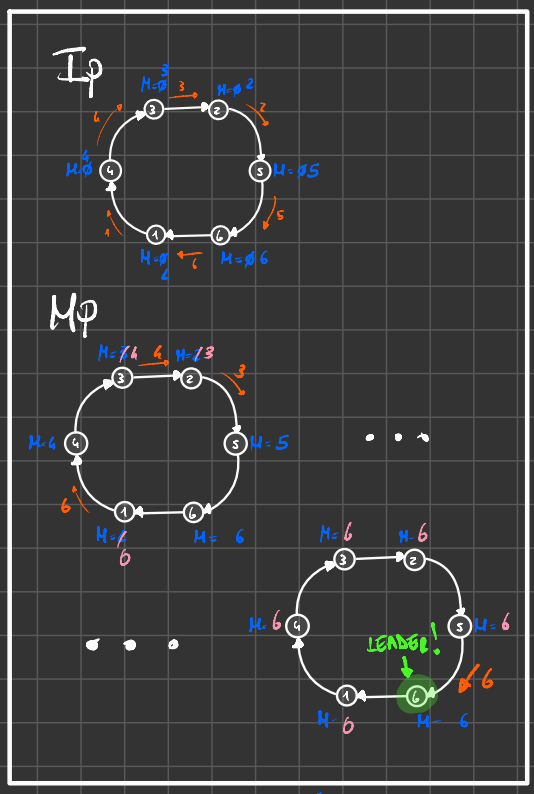
(// P is Processor ID)

Mp {RECEIVING J}

IF (J > M)
  M = J
  SEND M

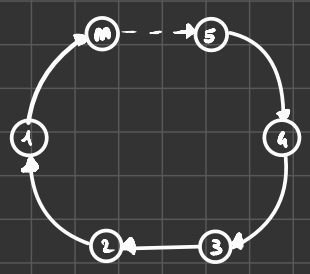
IF (M = P)
  PROCLAMATION

  STOP



NOTE → IGNORE PROCLAMATION

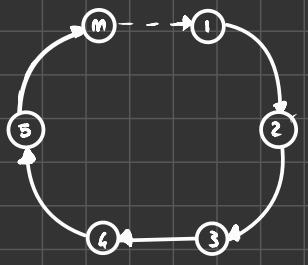
WORST CASE



- #MSGs $\underbrace{m}_{\#\text{MSGs of LEADER}} + (m-1) + (m-2) + \_ + 1 =$

$$= \sum_{i=1}^{m} i = \frac{m(m+1)}{2} \cdot \Theta(m^2)$$

BEST CASE



- #MSGs $\underbrace{m}_{\#\text{MSGs of LEADER}} + \underbrace{1 + \dots + 1}_{m} = 2m \cdot \Theta(m)$

AVG CASE

- The PROB. for a MSG. to make one step is $1/m$, so to complete the circle is $\sum_{i=1}^{m} i \cdot \mathbb{P}(\text{MAKING ONE STEP}) \leq \Theta(m \log m)$

⇒ $O(m \log m)$

ROUND BASED LEADER ELECTION

- A node become PASSIVE if it's value p is less then the one it receave
- A passive node only forwards messages

FRANKLIN ALGO (ON BIDIRECTED G)

- Processes send msgs in 2 directions
- At each round at least half Procs become PASSIVE (for every 2 near nodes one should win) ⇒ #ROUNDS $O(\log m)$
- In each round ACTIVE send 2 msgs, PASSIVE forward 2 msgs ⇒ #MSGs $= 2m \cdot O(\log m) = O(m \log m)$

27/09/21

ELECTION ON FAULTY SYSTEMS

- We assume that links are faulty free.
- We assume a TIMEOUT. (BOUNDED SYNCHRONICITY)
- ⇒ It follows that we can DETECT FAULTY PROCESS
- Assume a fully-connected topology.

IDEA

- If the leader fail, every process will try to replace it.

BULLY ALGO [GARCIA-MOLINA]

Detection PH.

- TIMEOUT reached while waiting LEADER's ping reply.

Preparation PH.

- P_i sends "ELECTION" msg to every $P_j \mid j > i$

Election PH.

- If no P_j reply in-TIME then I'M THE LEADER ⇒ Proclamation PH
- If $P_j \mid j > i$ receive P_i msg ⇒ P_j reply "YOU CANNOT BE THE LEADER" ⇒ P_j (IF NOT YET) goes to PREPARATION PH. and P_i goes to PROCLAMATION PH. (will wait a LEADER)

Proclamation PH.

- LEADER P_i LEADER send it's "ID" to all $P_j \mid j < i$
- Other P_s waits for LEADER PROCLAMATION MSG, if TIMEOUT expires ⇒ goto PROCLAMATION PH.

  LEADER FAILED DURING IT'S OWN PROCLAMATION

COMPLEXITY

- When P_i detects LEADER FAILURE it broadcast to his $(n-1)$ neighborhood, getting back at most $(n-1)$ replies
- WORST CASE:

$$\#MSGS_{worst} = O(n^3)$$

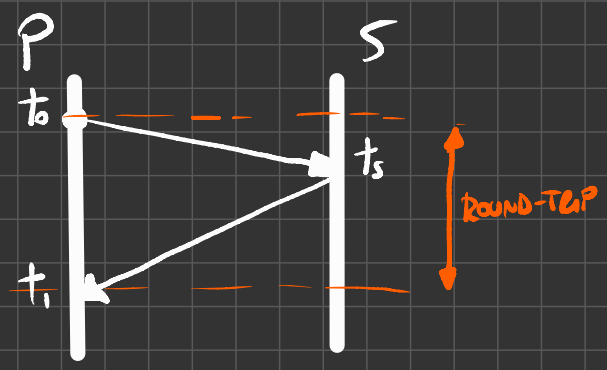
NOTE

- If a failed Processore is restarted it try to elect himself, so if it's the greatest ID proc., we will have 2 LEADERS, we should so consider EPOCHS to keep the latest elected one.

27/09/21

PHYSICAL CLOCKS

CHRISTIAN'S PHISICAL CLOCK



$$\Rightarrow t_1 = t_s + \frac{\text{ROUND-TRIP}}{2}$$

COMPUTER CLOCK

- Compute the average of the times of every processor; then each one set the average as it's TIME

BERKEG'S PHYSICAL CLOCK

- Same as COMPUTER CLOCK, but it consider ROUND-TRIP DELAYS TOO

LOGICAL CLOCKS

LAMPORT'S LOGICAL CLOCK

- Each P_i starts with a var. $L = \emptyset$
- BEFORE a LOCAL EVENT $L = L + 1$
- A msg m sent have L as TIMESTAMP
- When a msg m is received $L = \max(L, \text{TIMESTAMP}(m)) + 1$

RECEIVING m IS AN EVENT

VECTOR CLOCKS

- P_i start with a vector $V = [V_1, \_\_, V_n]$ inizialized to $\emptyset$
  - $V[i]$ contains #events timestamped by P_i
  - $V[j]$ $\forall j \neq i$ contains #events of P_j which have interacted with P_i
- BEFORE P_i LOCAL event: $V[i] = V[i] + 1$
- A msg sent has the whole vector V
- When a msg m is received from P_j $V[j] = \max(V[j], V_{RCV}[j])$ $\forall j$