Self-stabilizing Systems in Spite of Distributed Control

For Papers We Love

Non Self Stabilizing system

Once in a error state - it will remain in that forever

Problem when number of components grow - restart seem impossible

We need to add fault tolerance component

Good for synchronization, mutual exclusion,dining philosophers problems

History

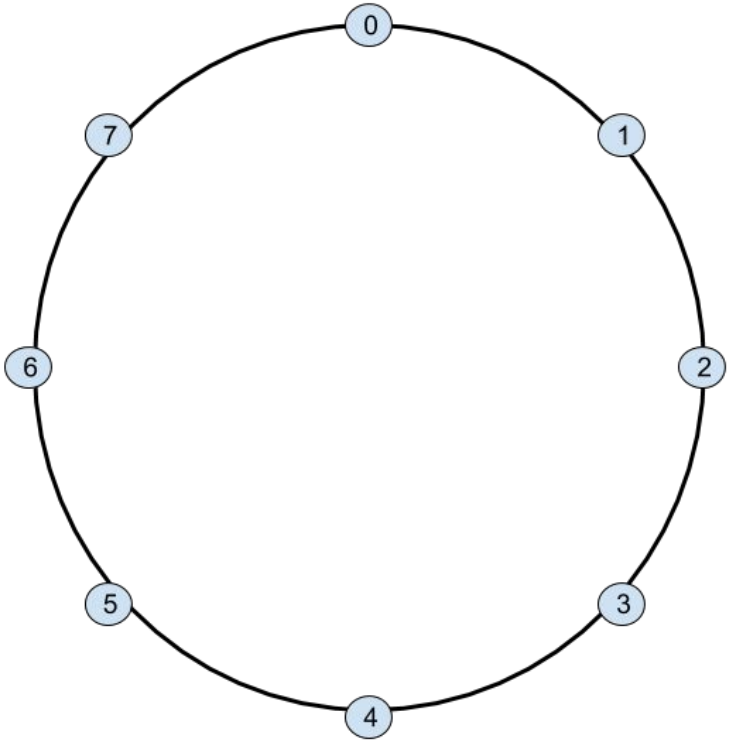
Self Stabilization in spite of Distributed Control *Springer-Verlag* (1973), 41-46

Errors found by C.S. Scholten

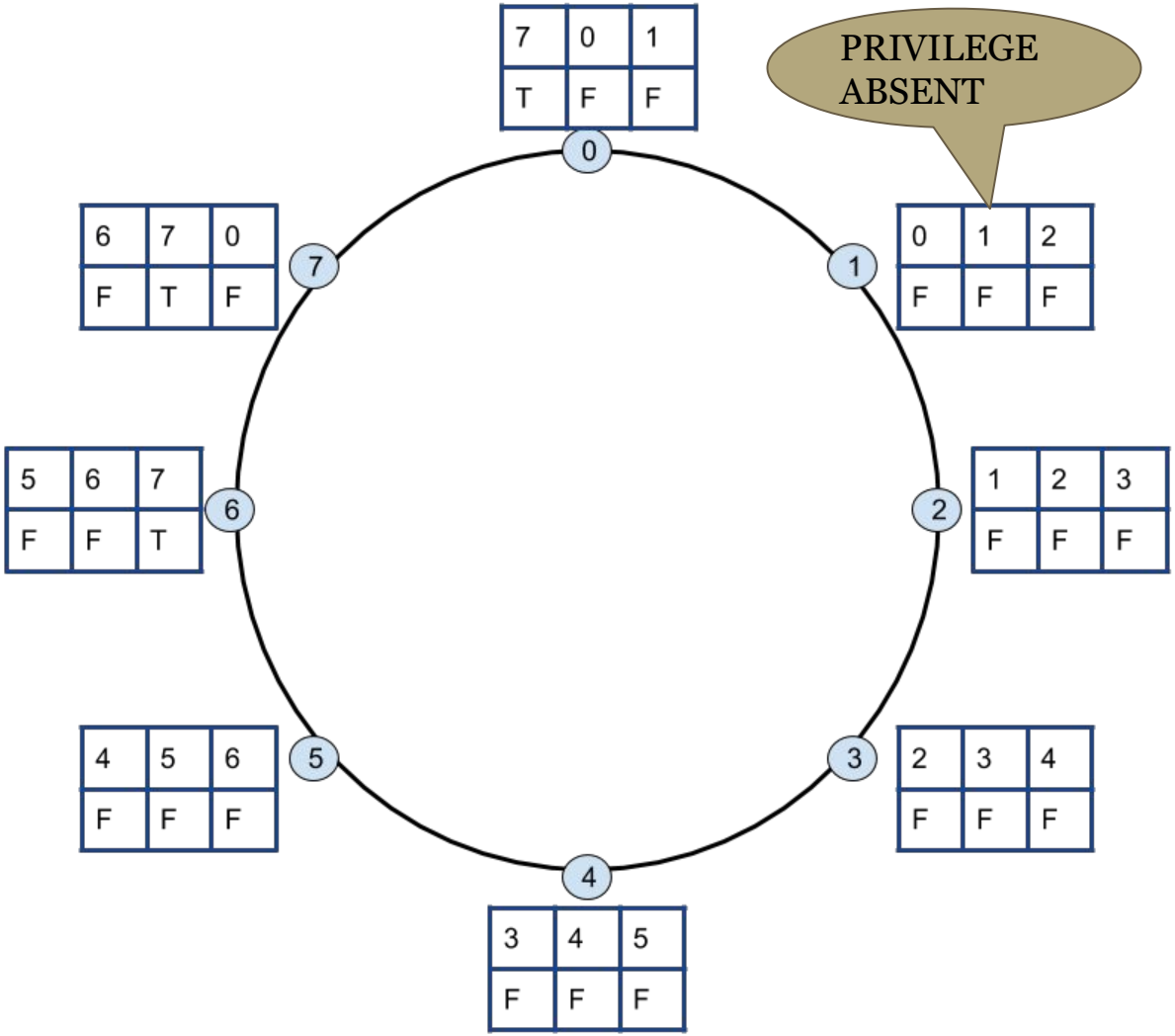
Self stabilizing in spite of Distributed Control *Communications of the ACM 17*, 1 (November 1974), 643–644.

A Belated proof on Self Stabilization *Distributed Computing 1*, 1 (January 1986), 5–6.

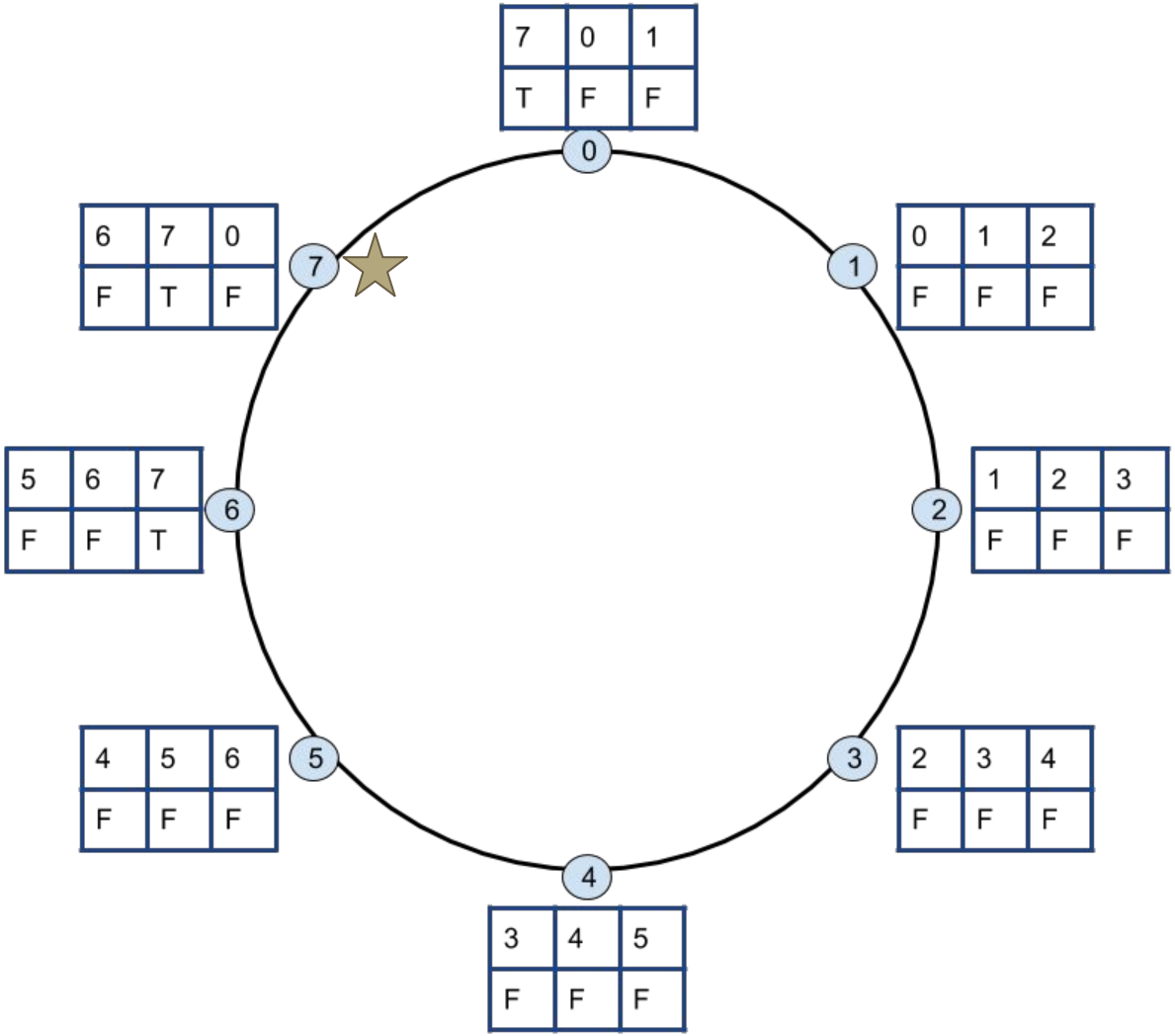
System



System


7 0 1
T F F
0
PRIVILEGE ABSENT
0 1 2
F F F
1
1 2 3
F F F
2
2 3 4
F F F
3
3 4 5
F F F
4
4 5 6
F F F
5
5 6 7
F F T
6
6 7 0
F T F
7

System


7 0 1
T F F
0
0 1 2
F F F
1
1 2 3
F F F
2
2 3 4
F F F
3
3 4 5
F F F
4
4 5 6
F F F
5
5 6 7
F F T
6
6 7 0
F T F
7

Legitimate State:

1. In each legitimate state one or more privileges will be present;
2. In each legitimate state each possible move will bring the system again in a legitimate state;
3. Each privilege must be present in at least one legitimate state;
4. For any pair of legitimate states there exists a sequence of moves transferring the system from the one into the other

Self Stabilizing

if and only if,

regardless of the initial state and regardless of the privilege selected

each time for the next move, at least one privilege will always be present and the system is guaranteed to find itself in a legitimate state

after a finite number of moves.

Solutions Proposed:

Solution with K state Machines

Solution with Four State Machines

Solution with Three State Machines

Solution with K State Machines

For 0th machine :

If L = S then S := (S + 1)mod K

for the other machines:

If L != S then S :=L

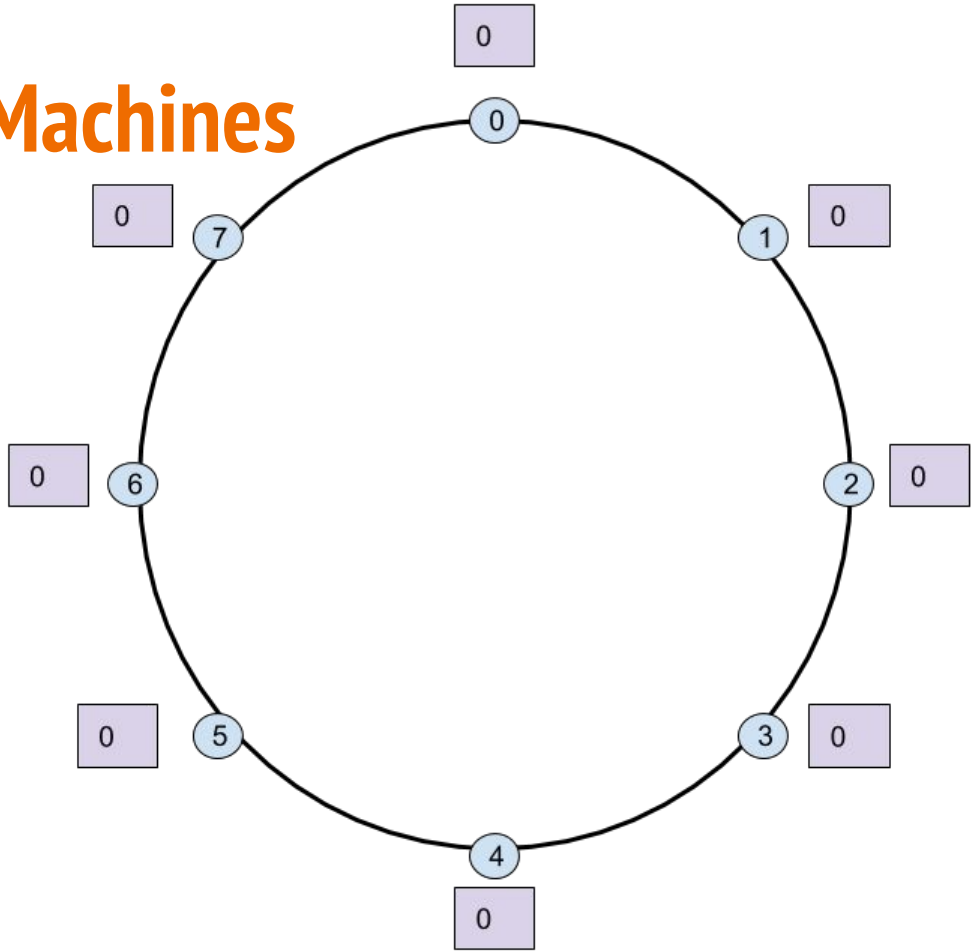
Solution with K State Machines

For 0th machine :

If L = S then S := (S + 1)mod K

for the other machines:

If L != S then S :=L



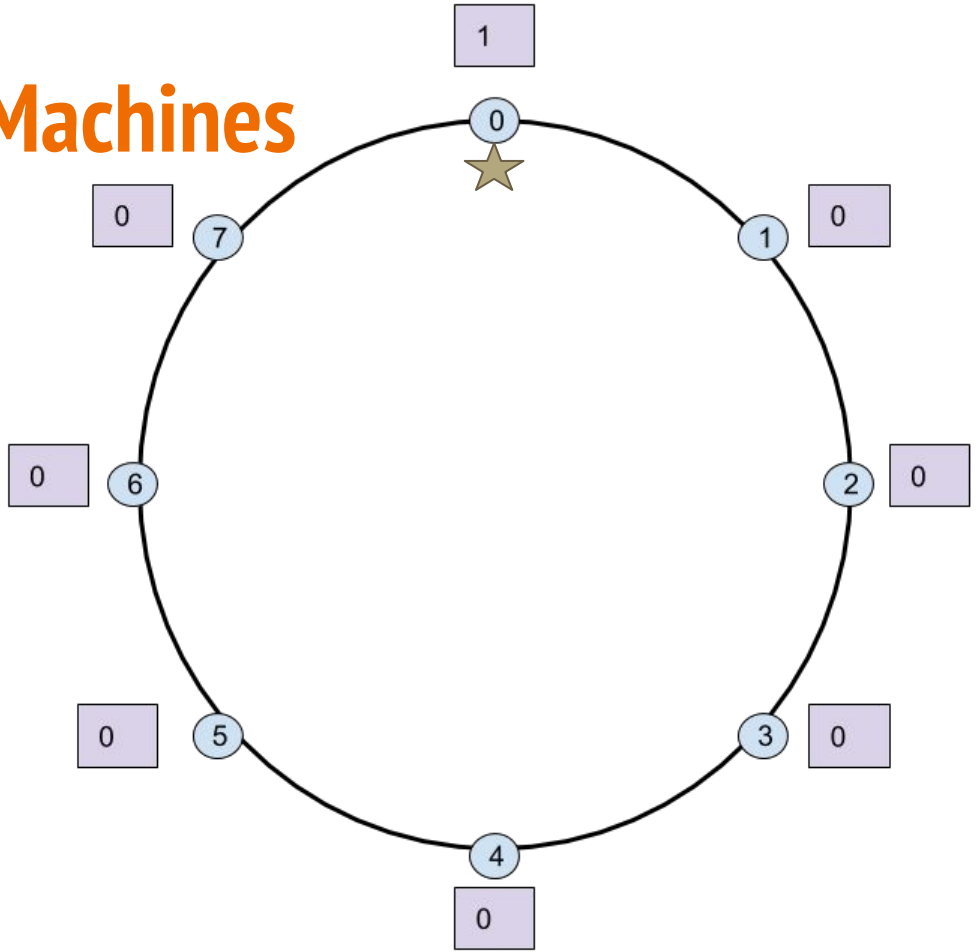
Solution with K State Machines

For 0th machine :

If L = S then S := (S + 1)mod K

for the other machines:

If L != S then S :=L



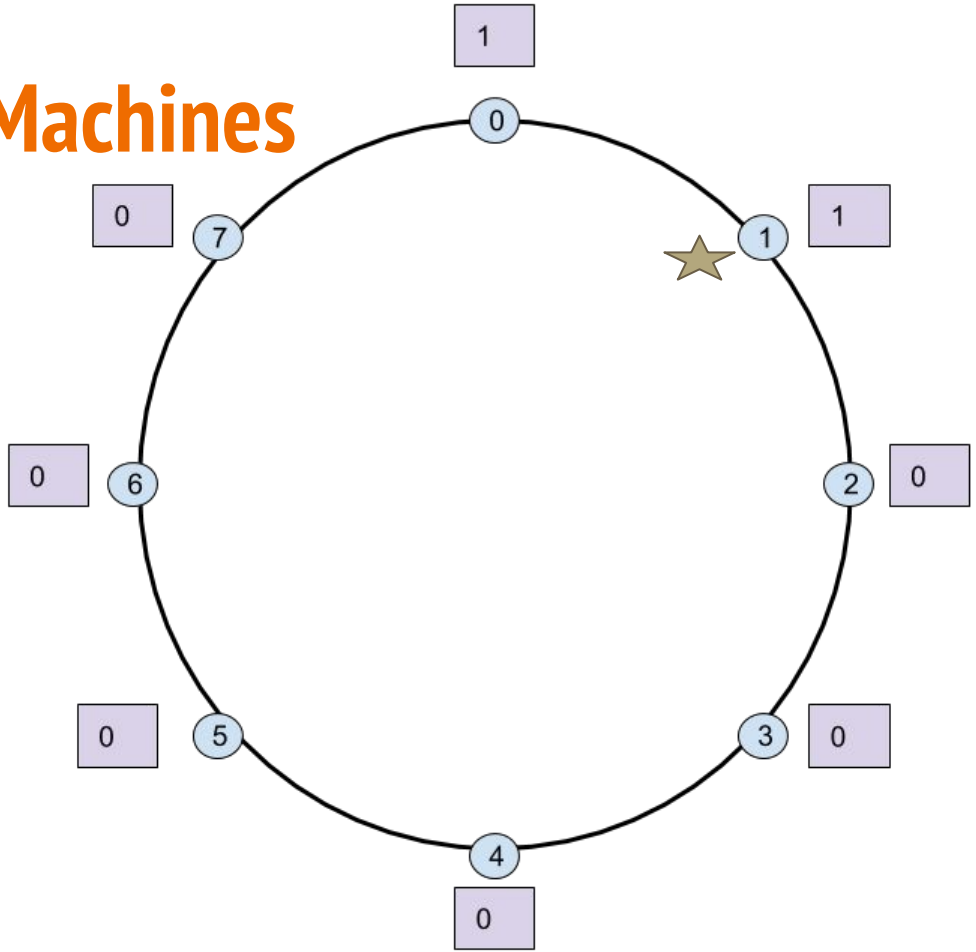
Solution with K State Machines

For 0th machine :

If L = S then S := (S + 1)mod K

for the other machines:

If L != S then S :=L



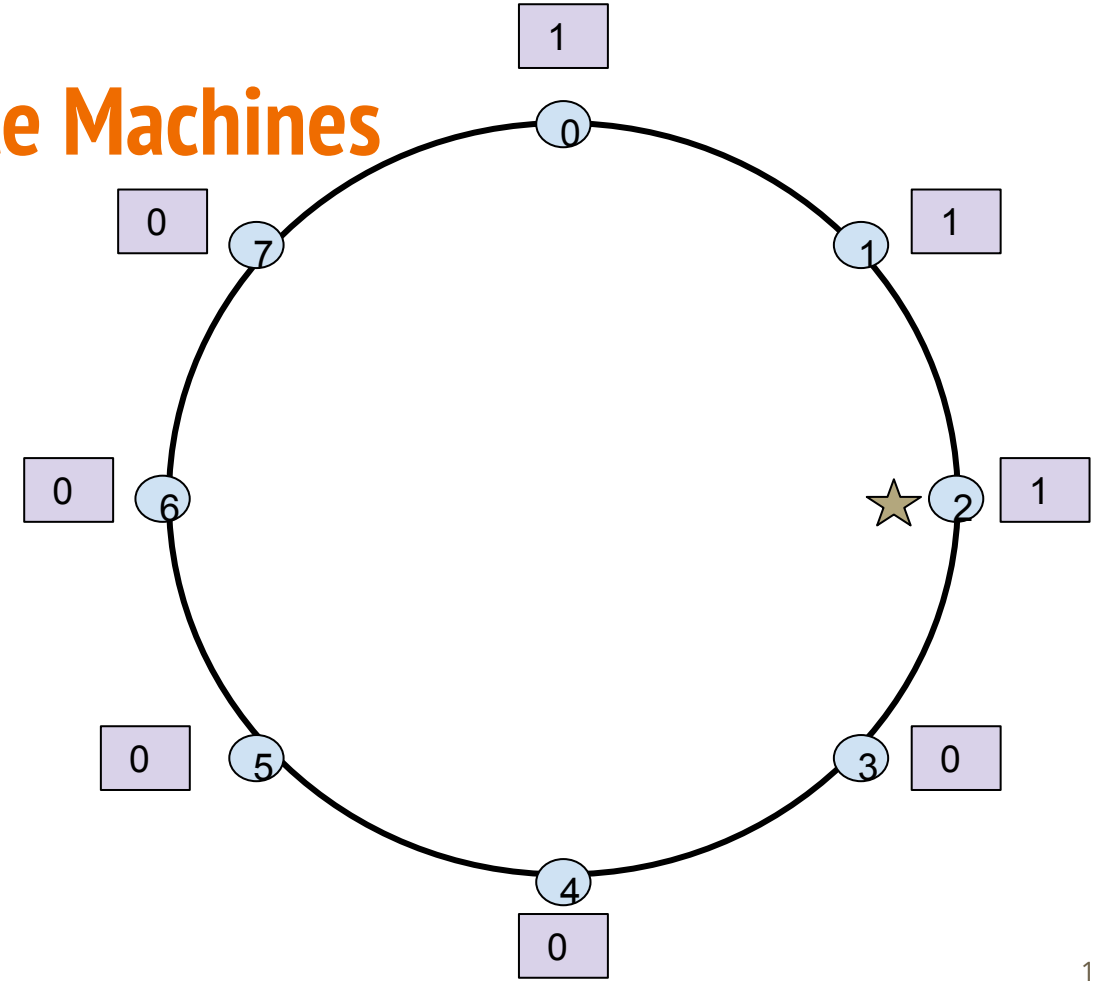
Solution with K State Machines

For 0th machine :

If L = S then S := (S + 1)mod K

for the other machines:

If L != S then S :=L



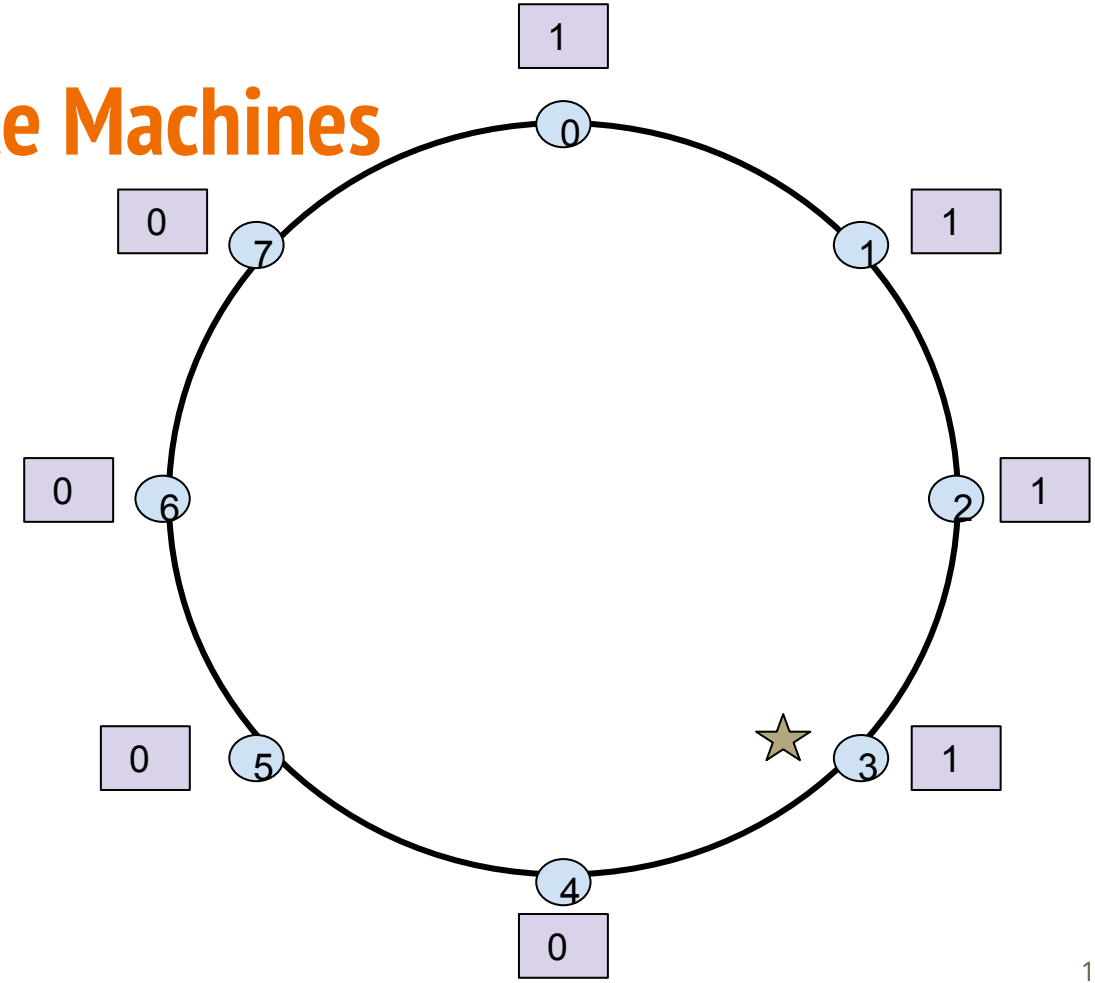
Solution with K State Machines

For 0th machine :

If L = S then S := (S + 1)mod K

for the other machines:

If L != S then S :=L



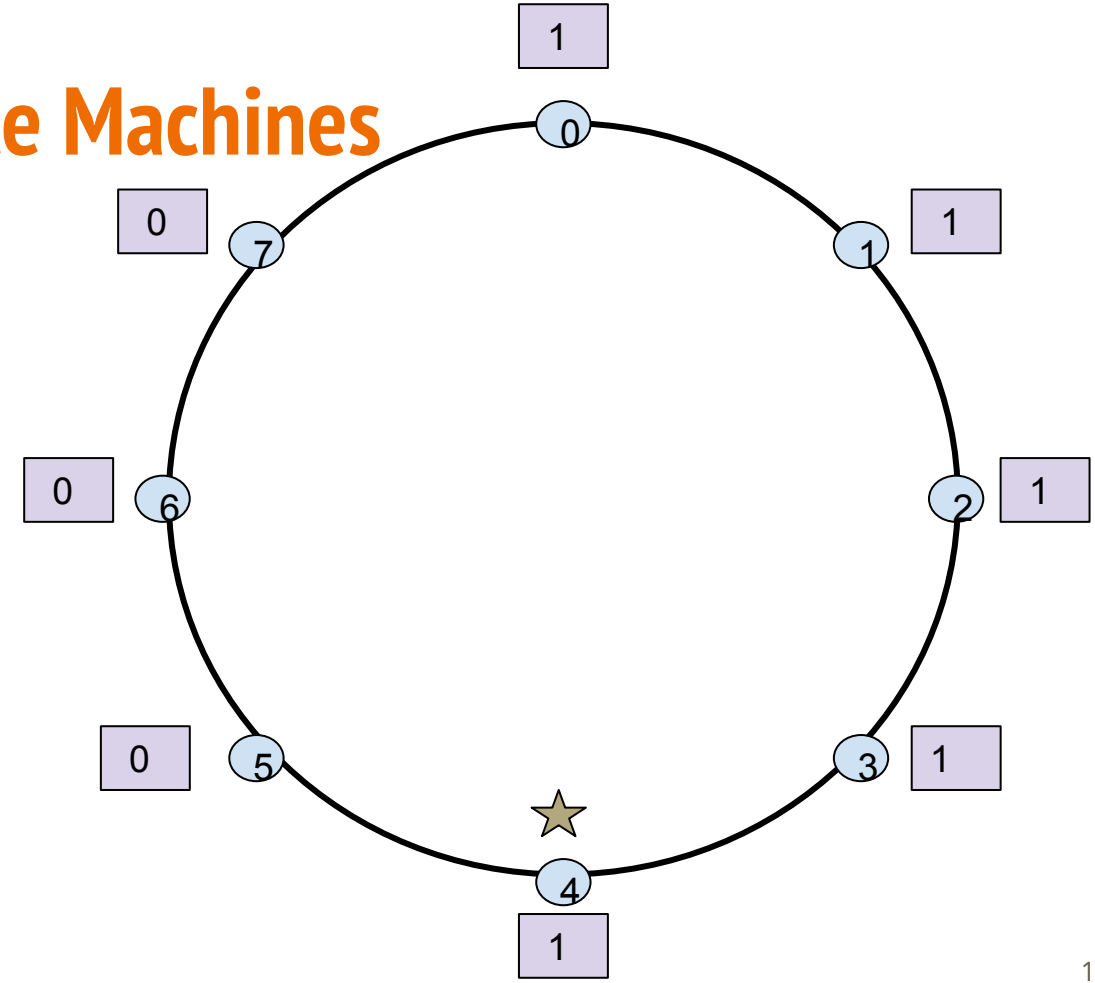
Solution with K State Machines

For 0th machine :

If L = S then S := (S + 1)mod K

for the other machines:

If L != S then S :=L



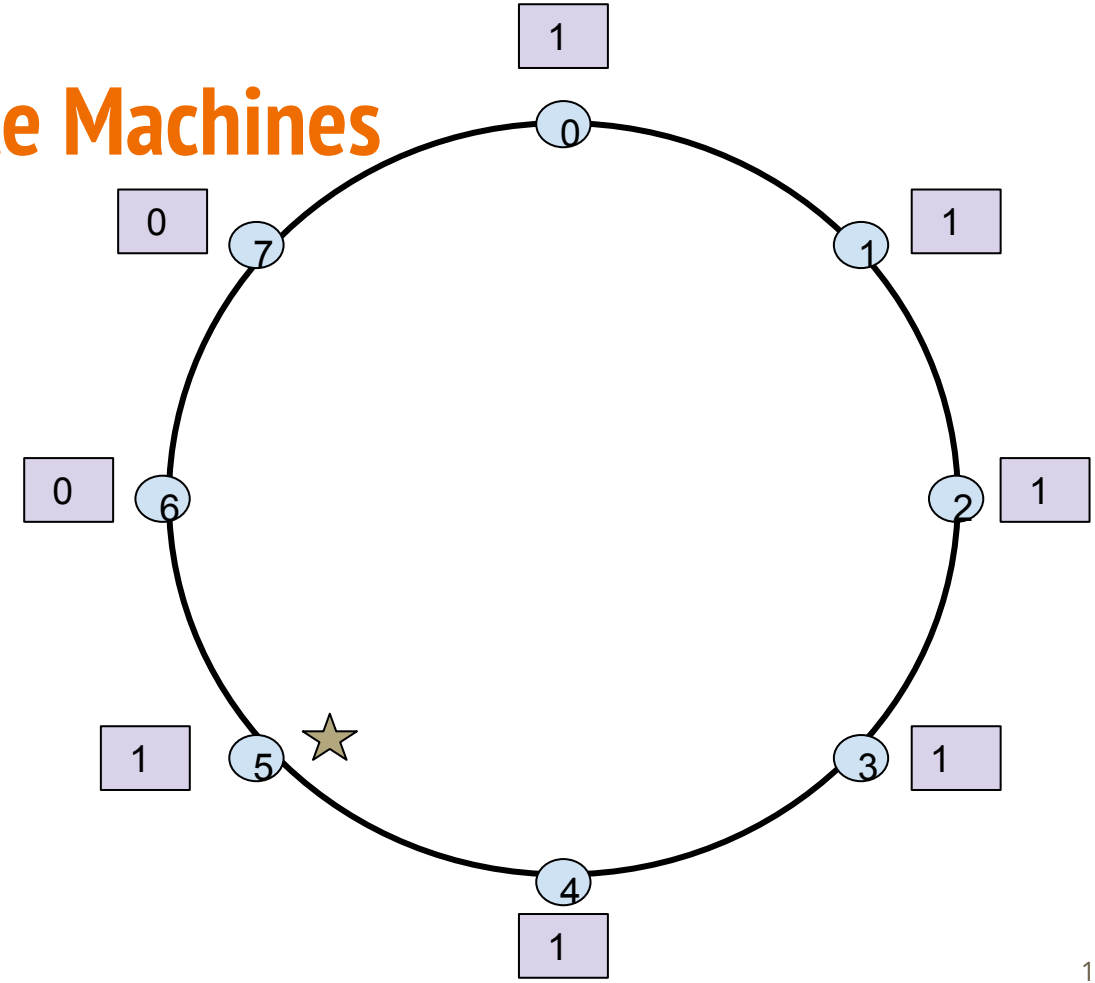
Solution with K State Machines

For 0th machine :

If L = S then S := (S + 1)mod K

for the other machines:

If L != S then S :=L



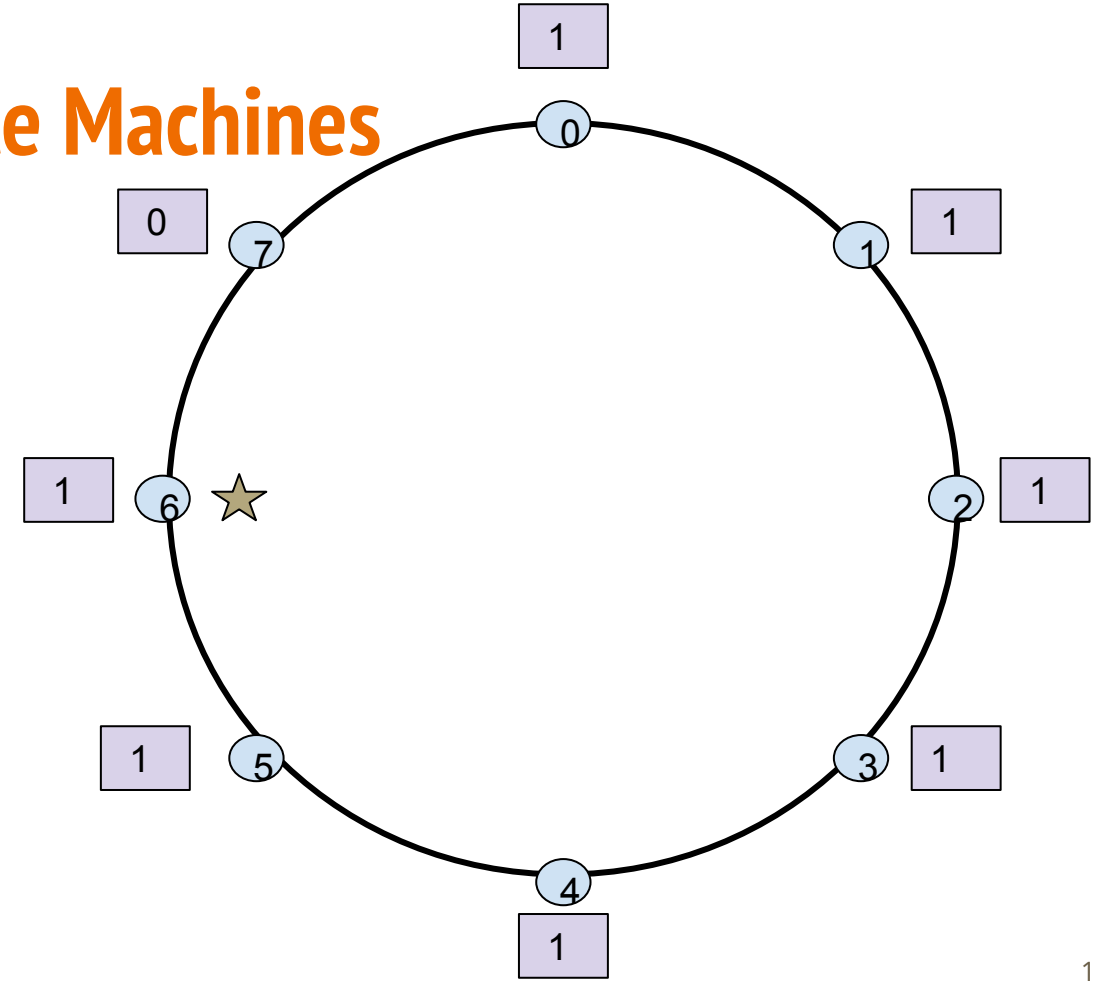
Solution with K State Machines

For 0th machine :

If L = S then S := (S + 1)mod K

for the other machines:

If L != S then S :=L



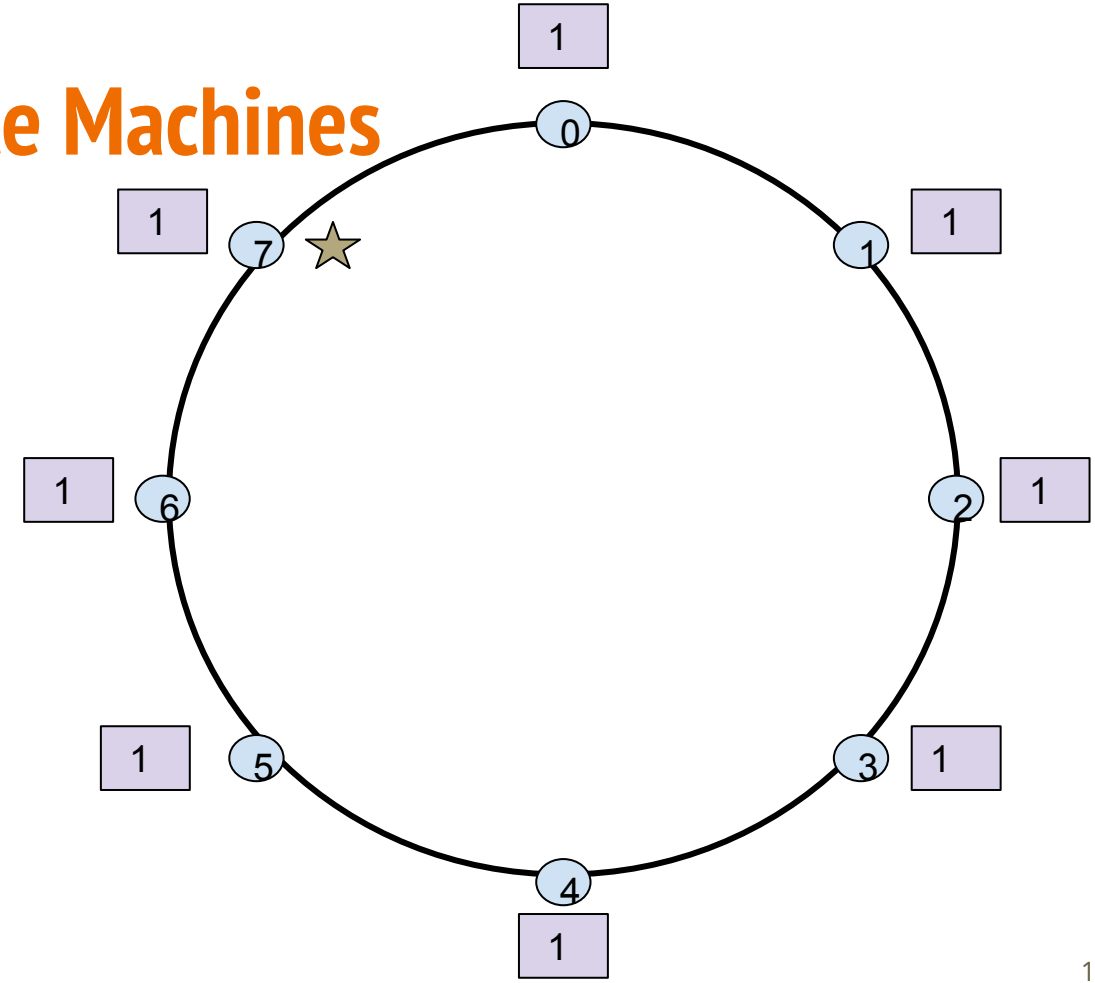
Solution with K State Machines

For 0th machine :

If L = S then S := (S + 1)mod K

for the other machines:

If L != S then S :=L



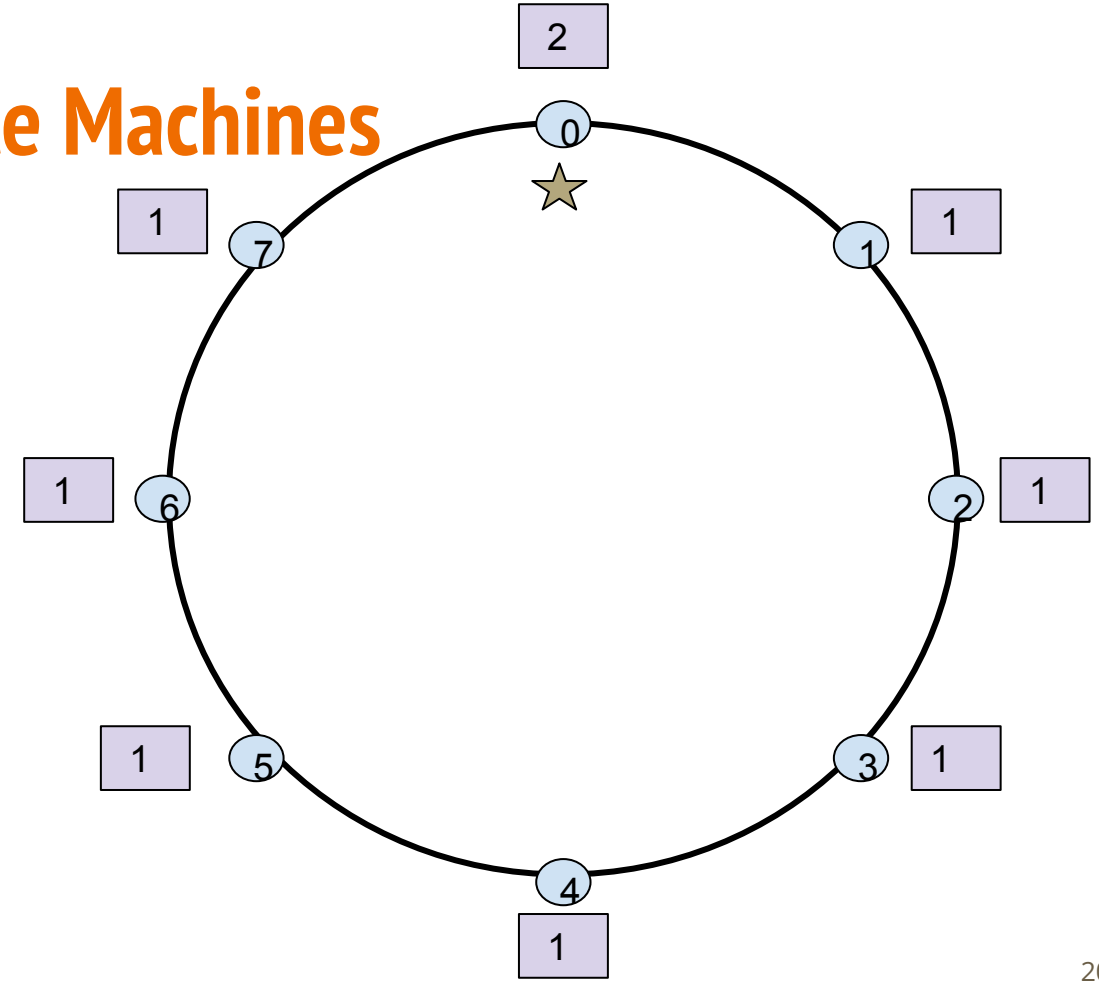
Solution with K State Machines

For 0th machine :

If L = S then S := (S + 1)mod K

for the other machines:

If L != S then S :=L



Solution with Four State Machine

Each machine has Two booleans :

- xS
- upS

For 0th Machine: upS = True

For Nth Machine: upS = False

Solution with Four State Machine

for the 0th machine: if xS = xR and non upR then xS : = non xS

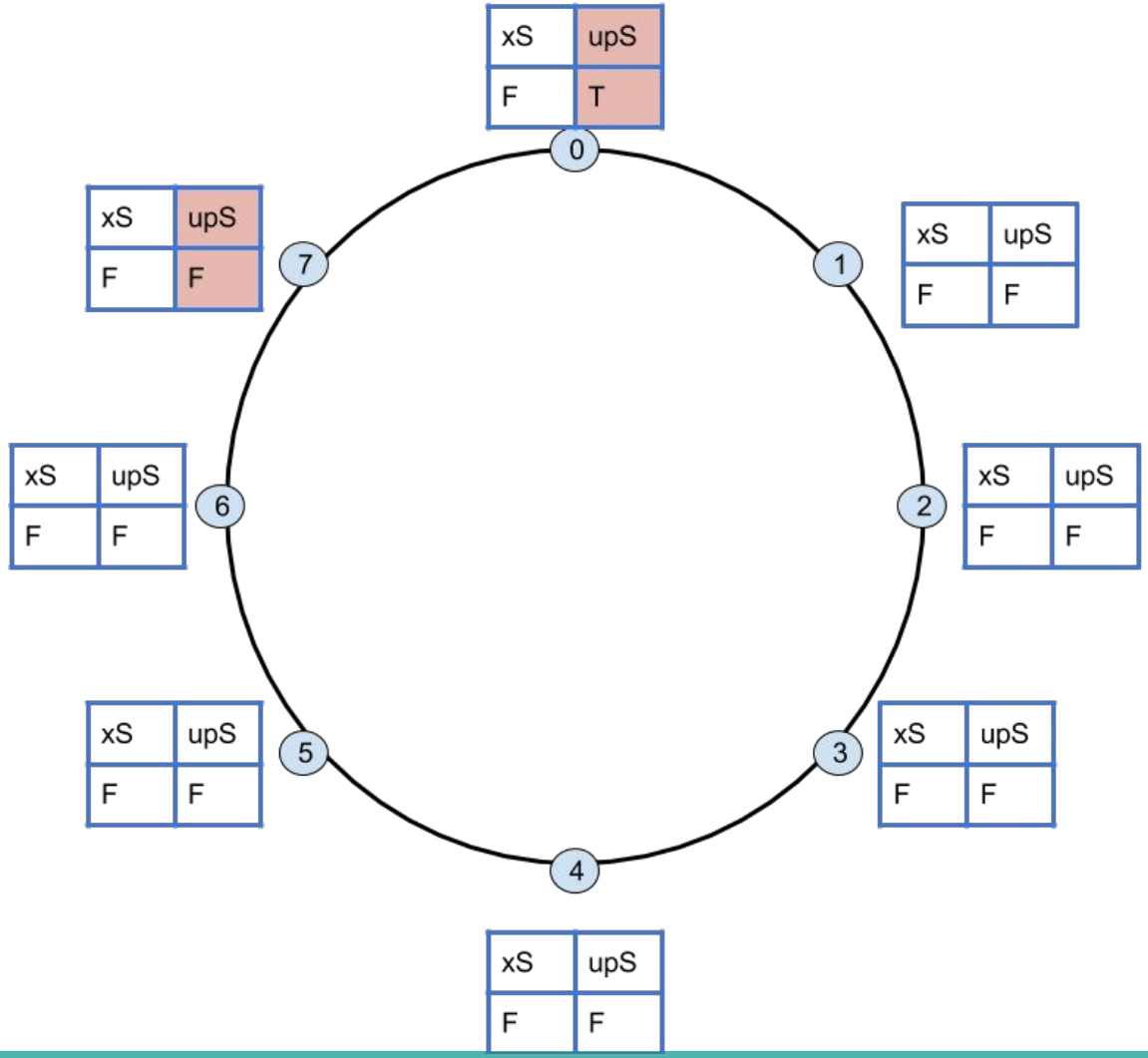
for the top machine: If xS != xL then xS := non xS

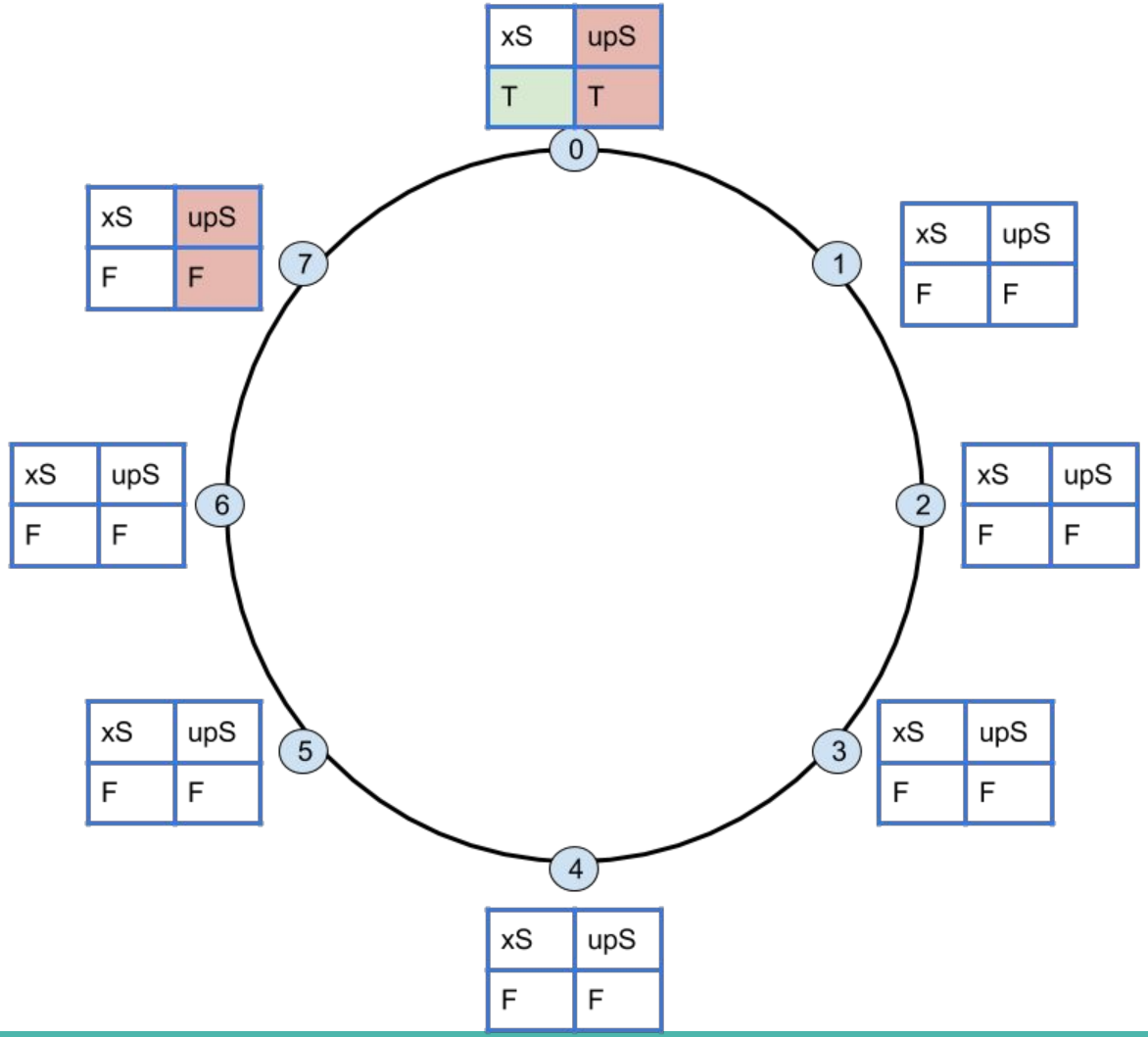
for the other machines:

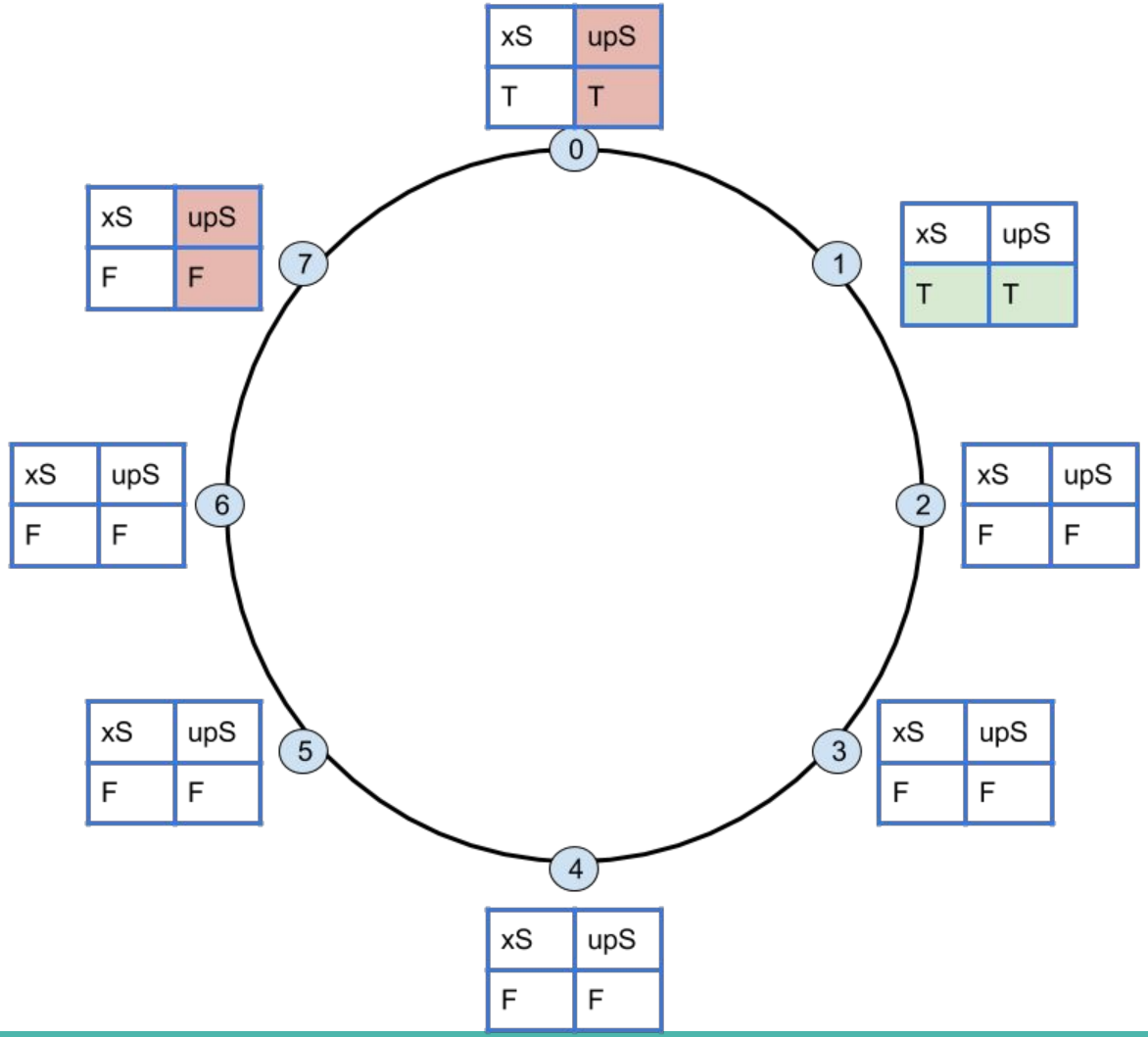
if xS != xL then xS := non xS;

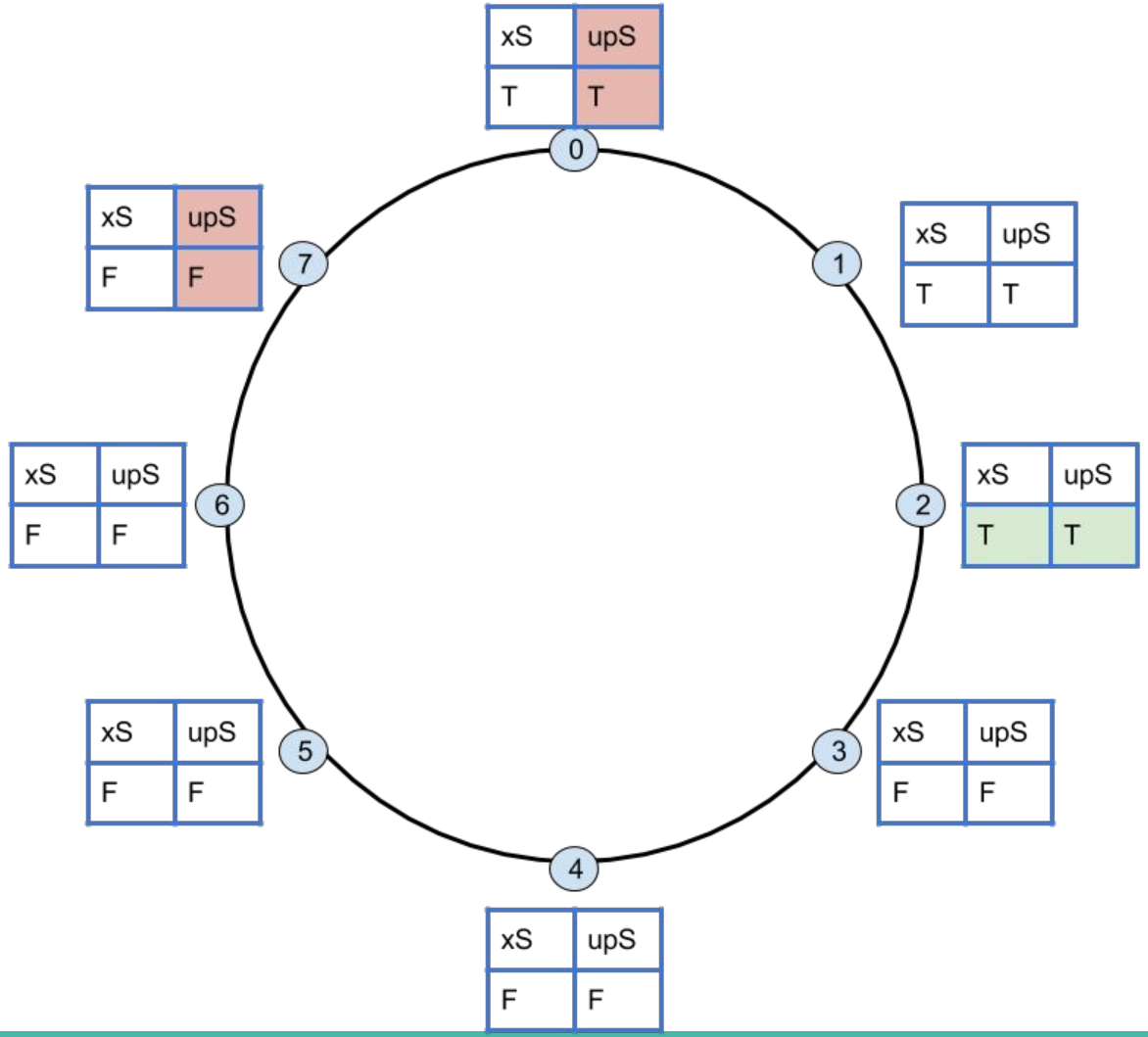
upS := true;

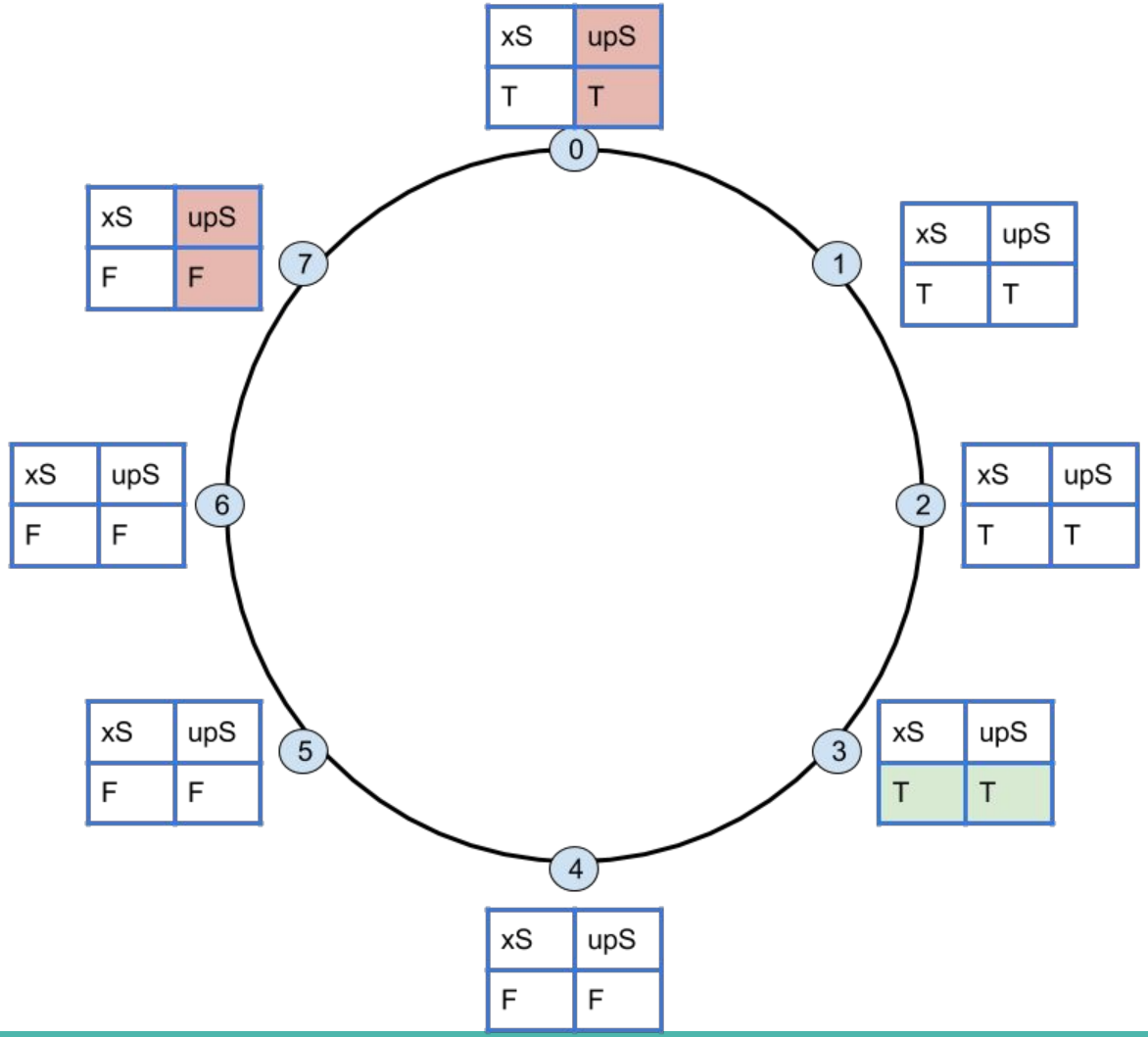
if xS = xR and upS and non upR then upS := false

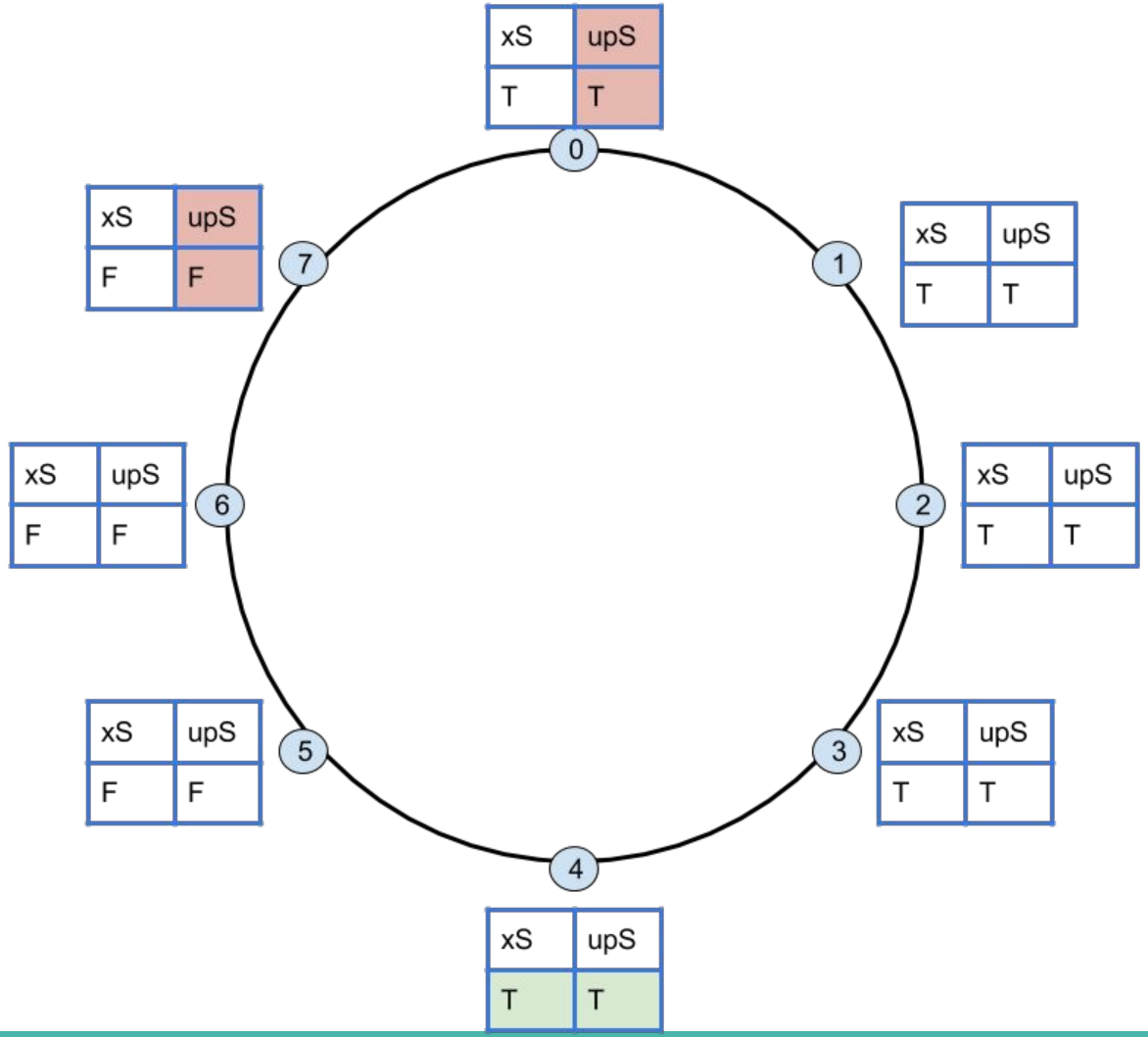

xS
upS
F
T
0
xS
upS
F
F
1
xS
upS
F
F
2
xS
upS
F
F
3
xS
upS
F
F
4
xS
upS
F
F
5
xS
upS
F
F
6
xS
upS
F
F
7

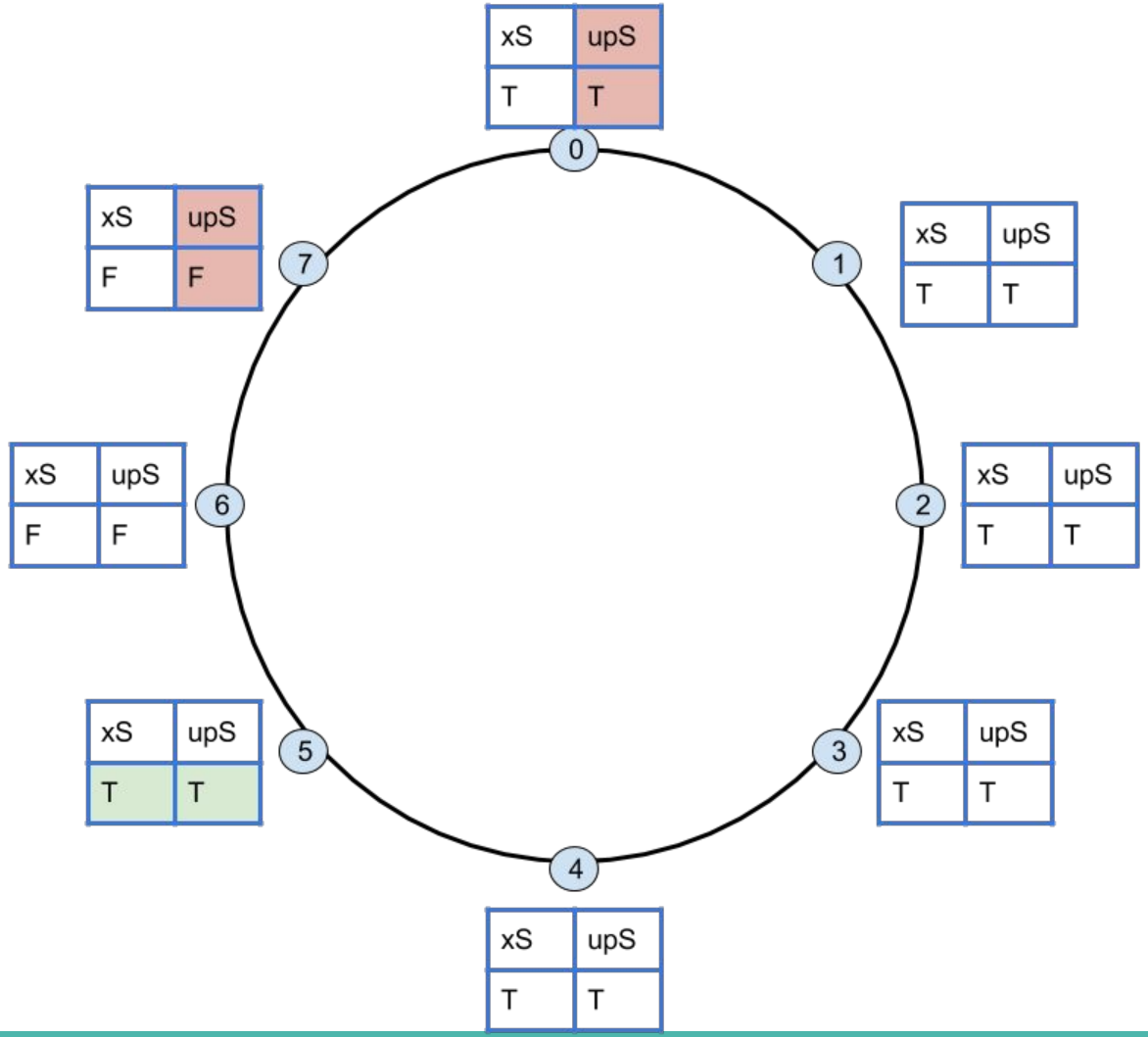

xS
upS
T
T
0
xS
upS
F
F
1
xS
upS
F
F
2
xS
upS
F
F
3
xS
upS
F
F
4
xS
upS
F
F
5
xS
upS
F
F
6
xS
upS
F
F
7

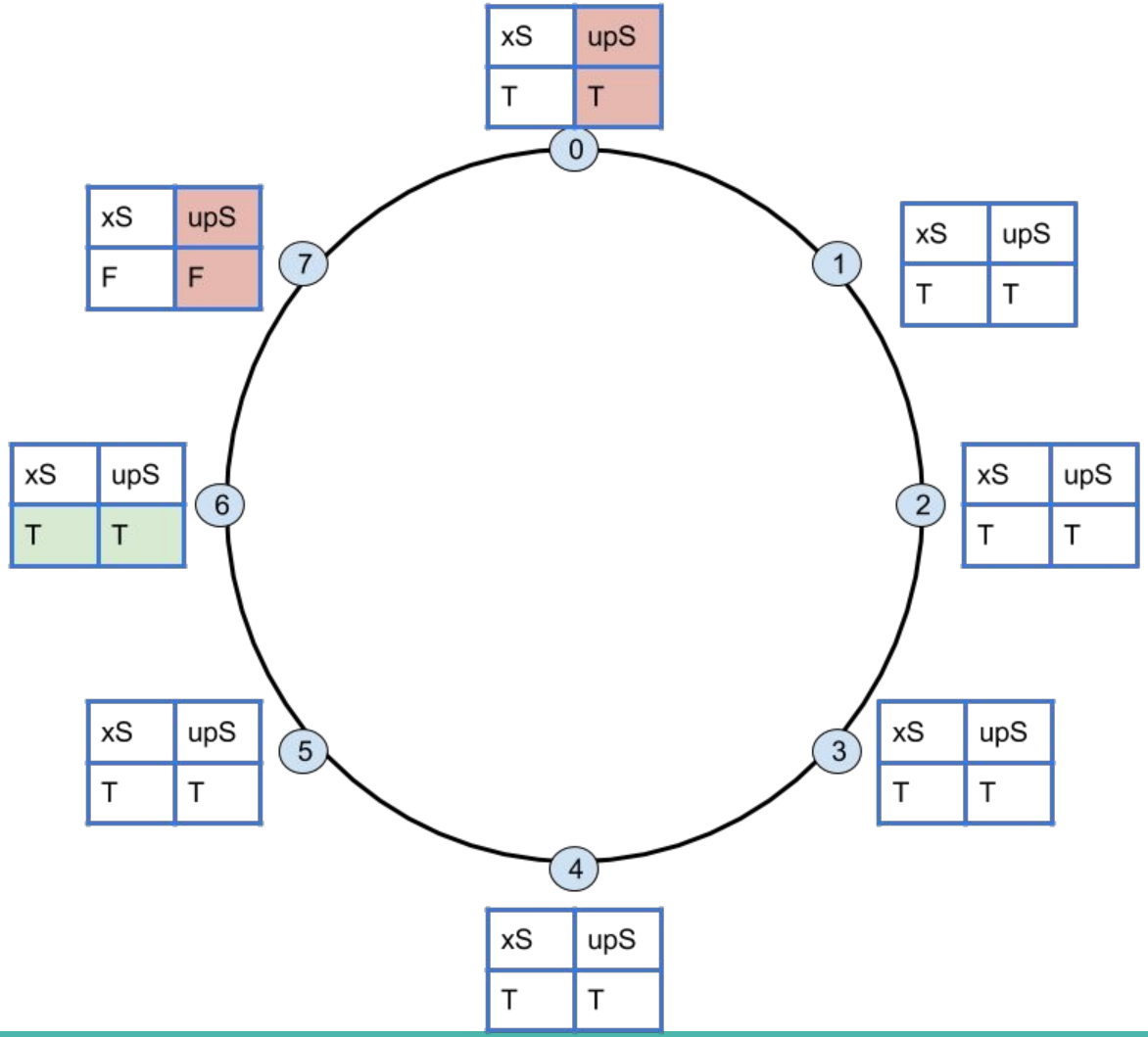

xS
upS
T
T
0
xS
upS
T
T
1
xS
upS
F
F
2
xS
upS
F
F
3
xS
upS
F
F
4
xS
upS
F
F
5
xS
upS
F
F
6
xS
upS
F
F
7

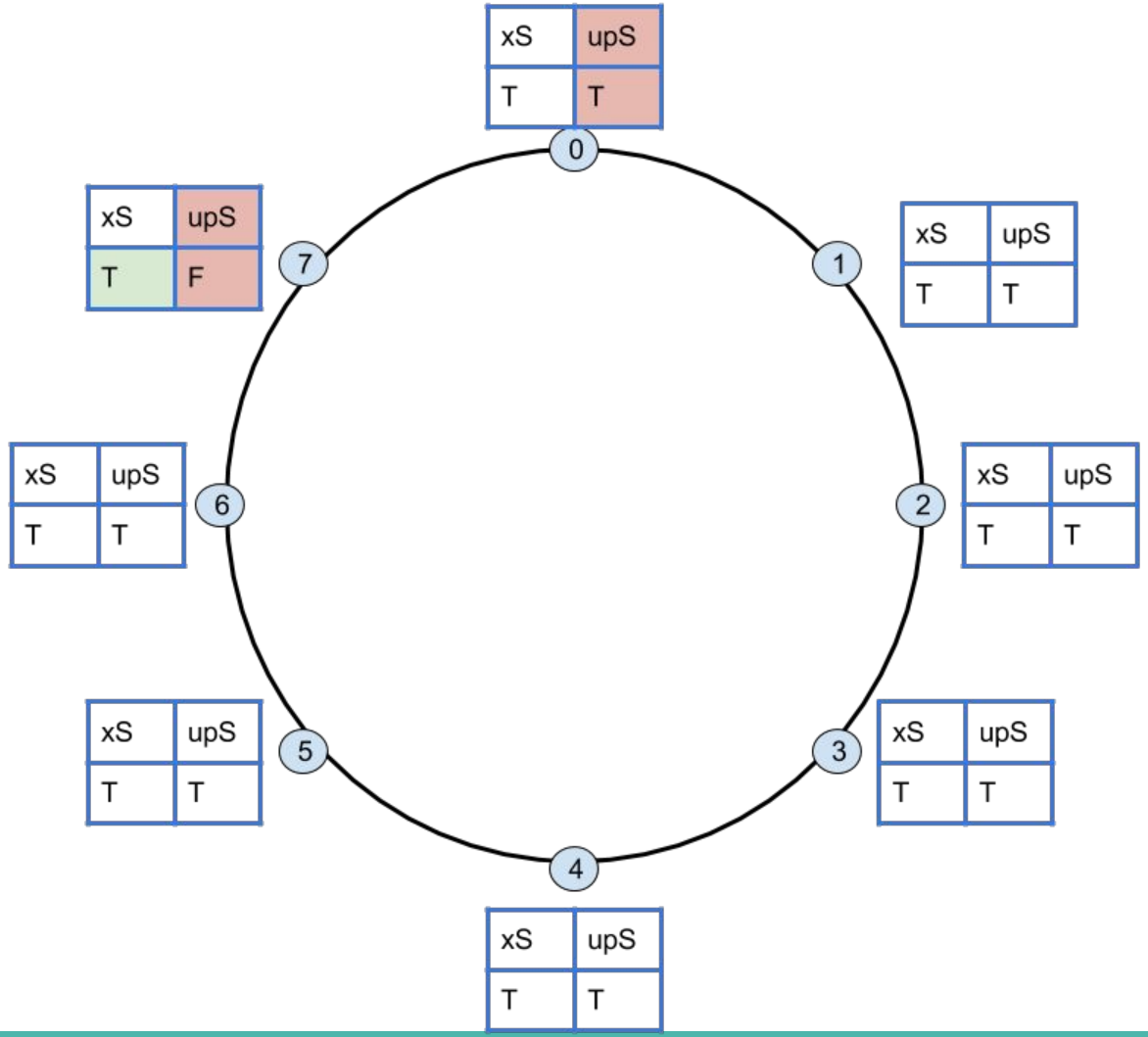

xS
upS
T
T
0
xS
upS
T
T
1
xS
upS
T
T
2
xS
upS
F
F
3
xS
upS
F
F
4
xS
upS
F
F
5
xS
upS
F
F
6
xS
upS
F
F
7

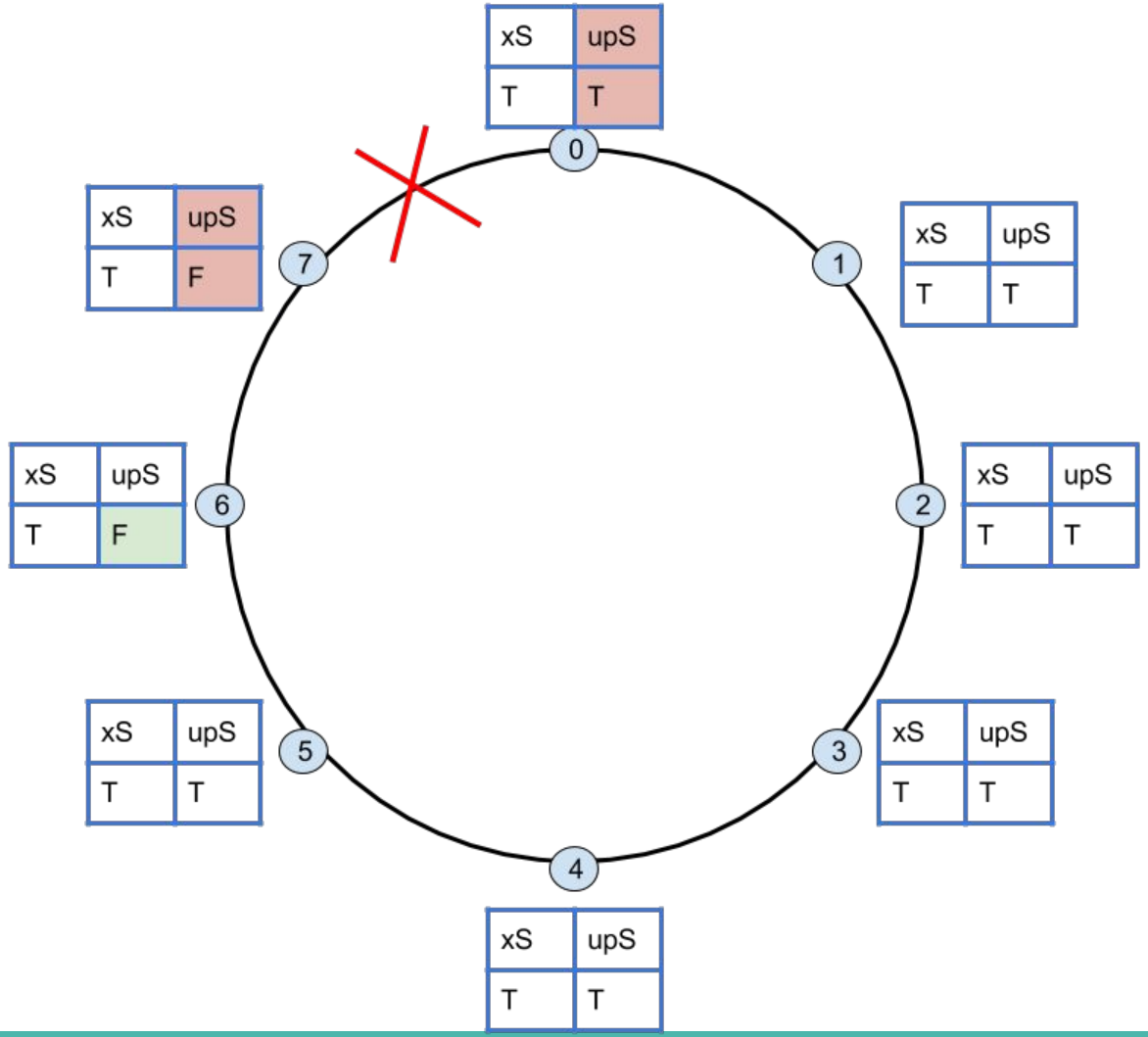

xS
upS
T
T
0
xS
upS
T
T
1
xS
upS
T
T
2
xS
upS
T
T
3
xS
upS
F
F
4
xS
upS
F
F
5
xS
upS
F
F
6
xS
upS
F
F
7

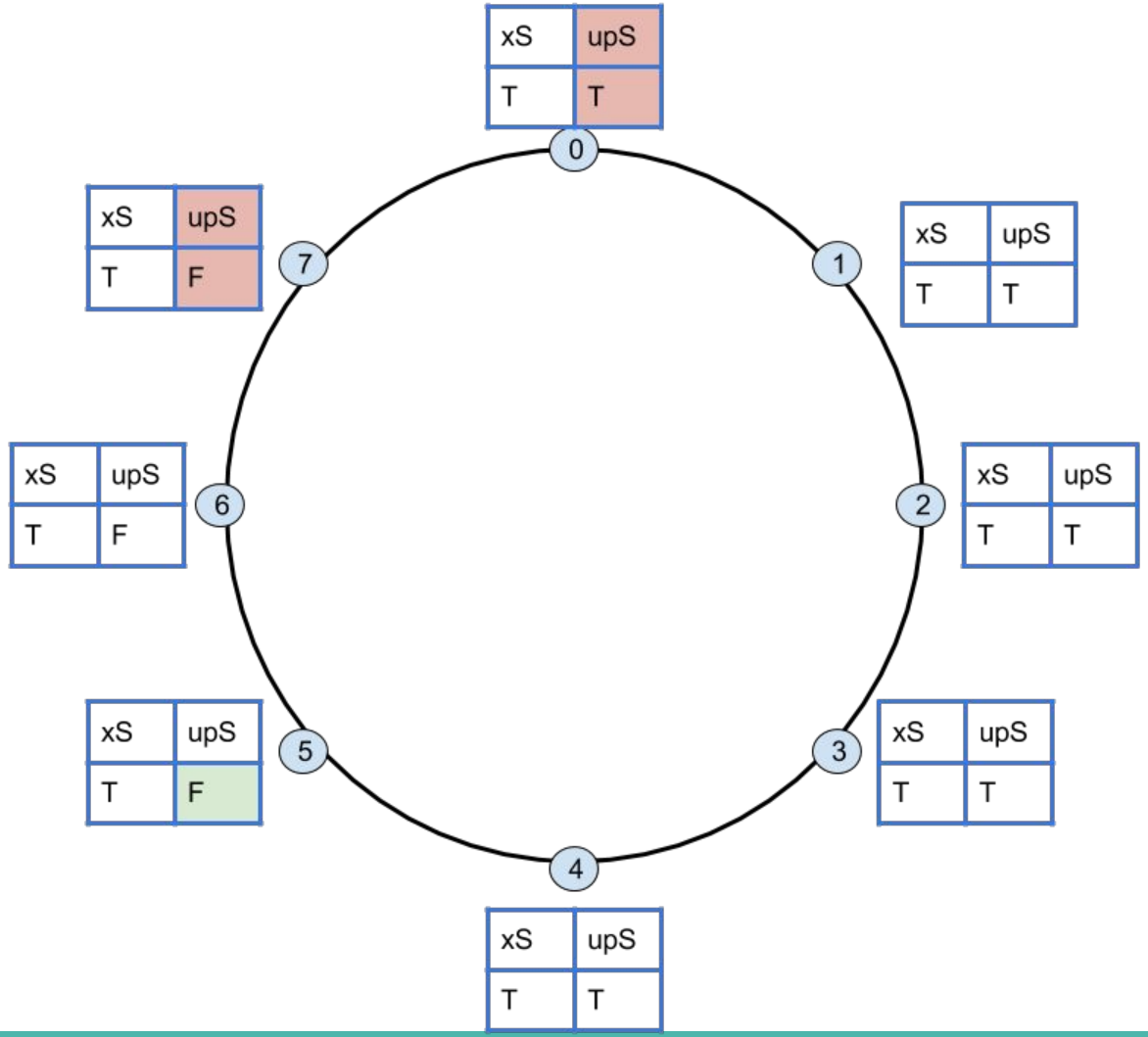

xS
upS
T
T
0
xS
upS
T
T
1
xS
upS
T
T
2
xS
upS
T
T
3
xS
upS
T
T
4
xS
upS
F
F
5
xS
upS
F
F
6
xS
upS
F
F
7

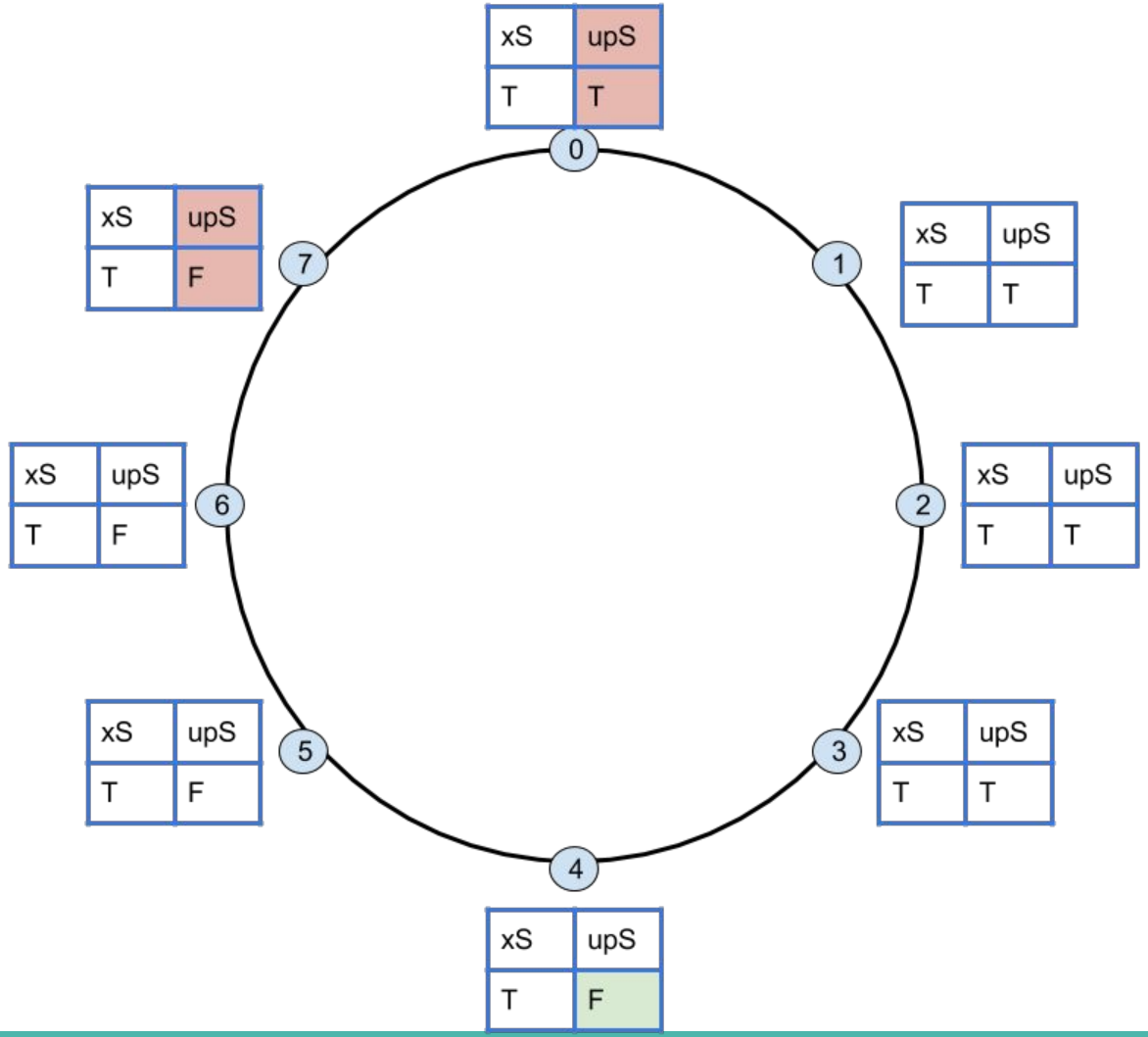

xS
upS
T
T
0
xS
upS
T
T
1
xS
upS
T
T
2
xS
upS
T
T
3
xS
upS
T
T
4
xS
upS
T
T
5
xS
upS
F
F
6
xS
upS
F
F
7

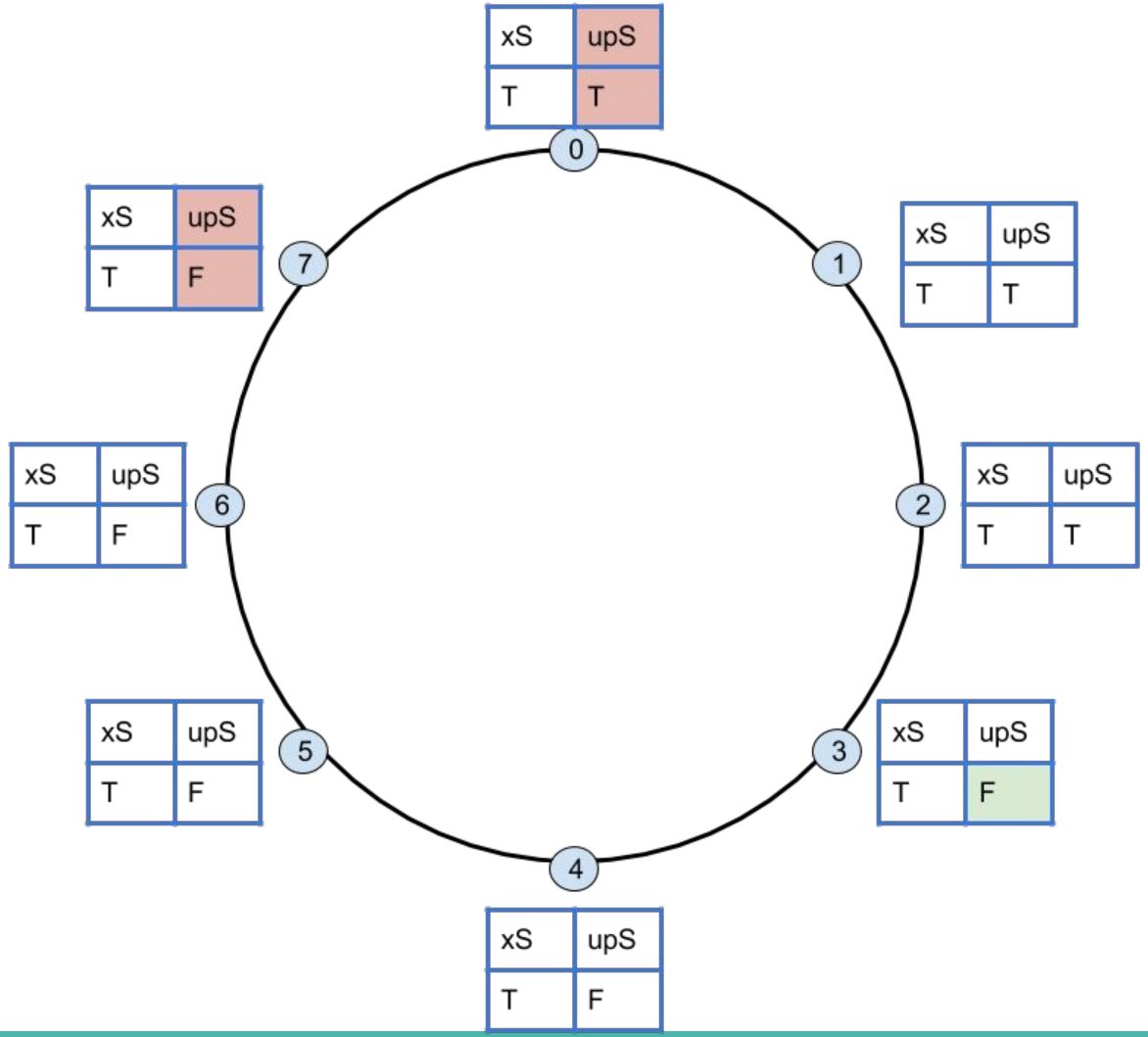

xS
upS
T
T
0
xS
upS
T
T
1
xS
upS
T
T
2
xS
upS
T
T
3
xS
upS
T
T
4
xS
upS
T
T
5
xS
upS
T
T
6
xS
upS
F
F
7

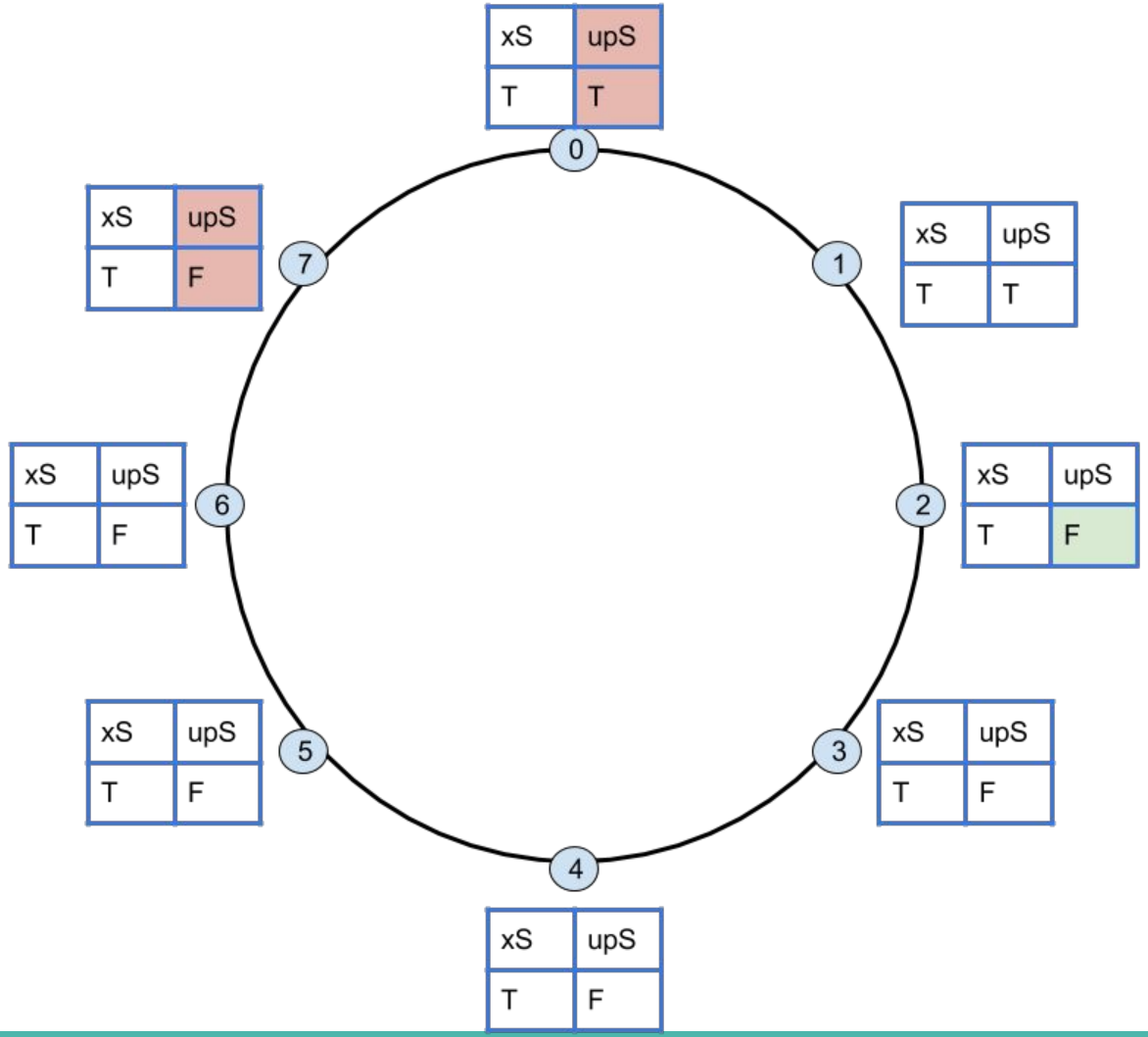

xS
upS
T
T
0
xS
upS
T
T
1
xS
upS
T
T
2
xS
upS
T
T
3
xS
upS
T
T
4
xS
upS
T
T
5
xS
upS
T
T
6
xS
upS
T
F
7

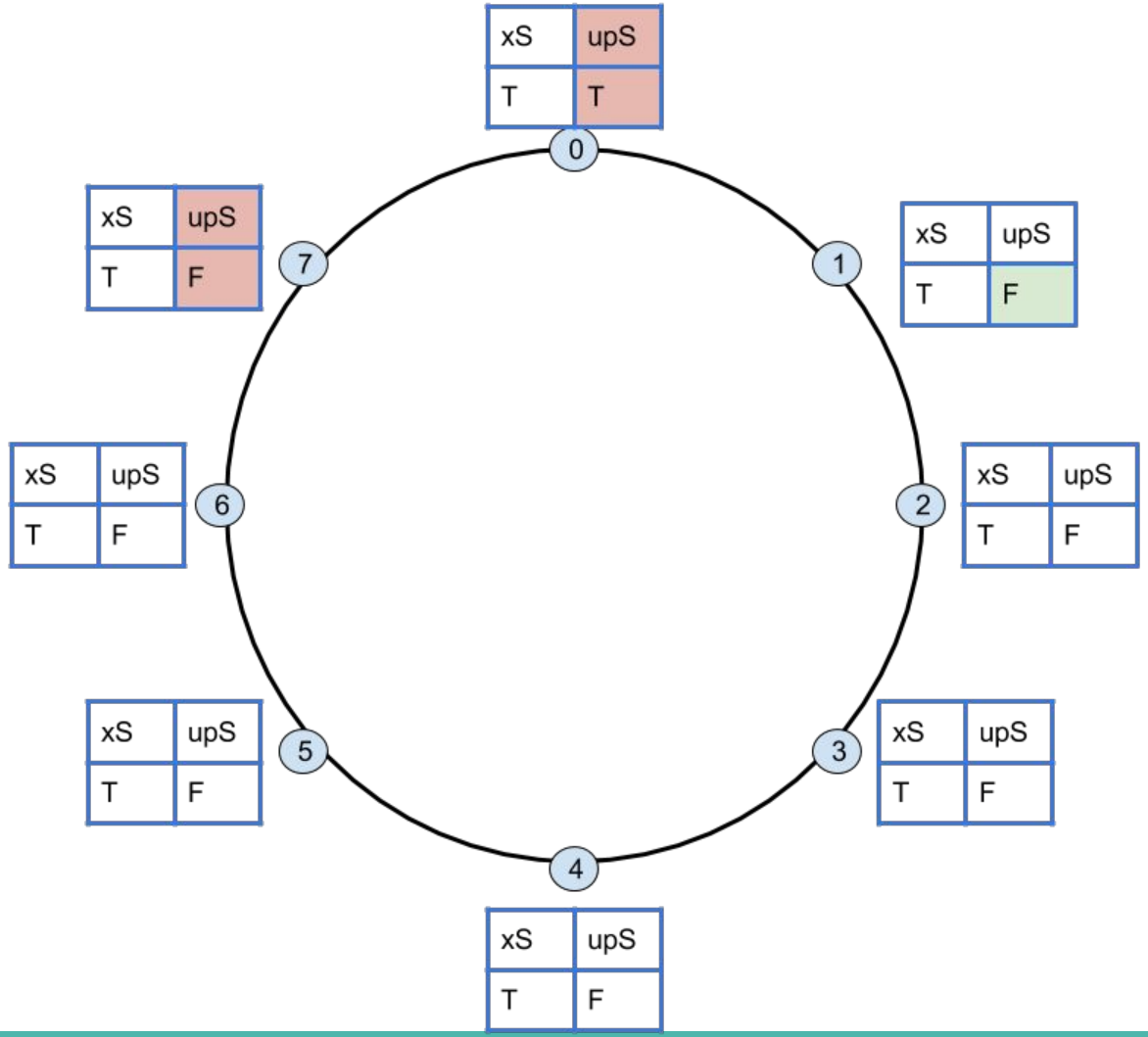

xS
upS
T
T
0
xS
upS
T
T
1
xS
upS
T
T
2
xS
upS
T
T
3
xS
upS
T
T
4
xS
upS
T
T
5
xS
upS
T
F
6
xS
upS
T
F
7

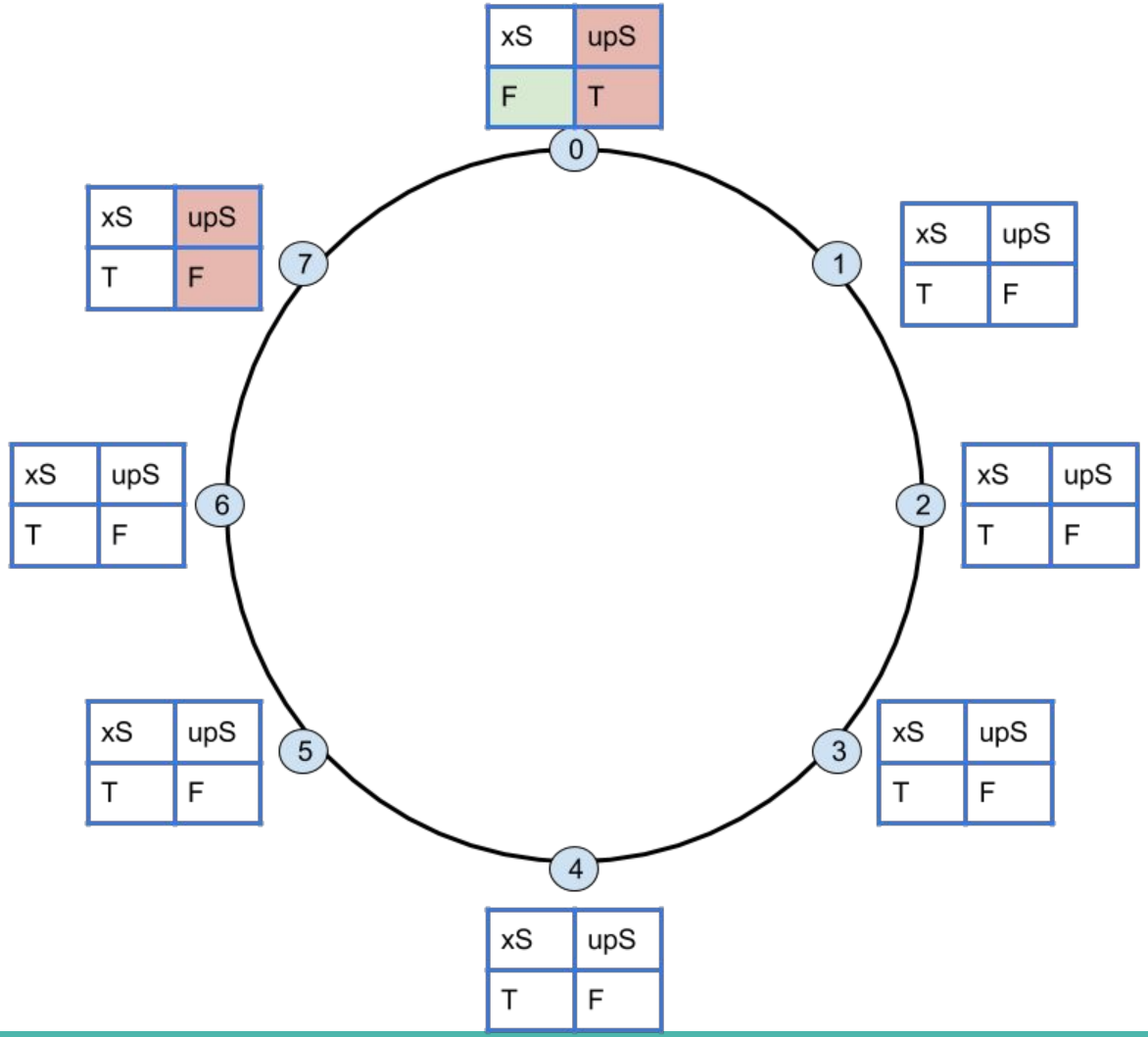

xS
upS
T
T
0
xS
upS
T
T
1
xS
upS
T
T
2
xS
upS
T
T
3
xS
upS
T
T
4
xS
upS
T
F
5
xS
upS
T
F
6
xS
upS
T
F
7


xS
upS
T
T
0
xS
upS
T
T
1
xS
upS
T
T
2
xS
upS
T
T
3
xS
upS
T
F
4
xS
upS
T
F
5
xS
upS
T
F
6
xS
upS
T
F
7


xS upS T T
0
xS upS T T
1
xS upS T T
2
xS upS T F
3
xS upS T F
4
xS upS T F
5
xS upS T F
6
xS upS T F
7


xS upS
T T
0
xS upS
T T
1
xS upS
T F
2
xS upS
T F
3
xS upS
T F
4
xS upS
T F
5
xS upS
T F
6
xS upS
T F
7


xS
upS
T
T
0
xS
upS
T
F
1
xS
upS
T
F
2
xS
upS
T
F
3
xS
upS
T
F
4
xS
upS
T
F
5
xS
upS
T
F
6
xS
upS
T
F
7


xS
upS
F
T
0
xS
upS
T
F
1
xS
upS
T
F
2
xS
upS
T
F
3
xS
upS
T
F
4
xS
upS
T
F
5
xS
upS
T
F
6
xS
upS
T
F
7

Solution with Three State Machines

for the 0th machine:

if (S + 1) mod 3 = R then S := (S - 1) mod 3

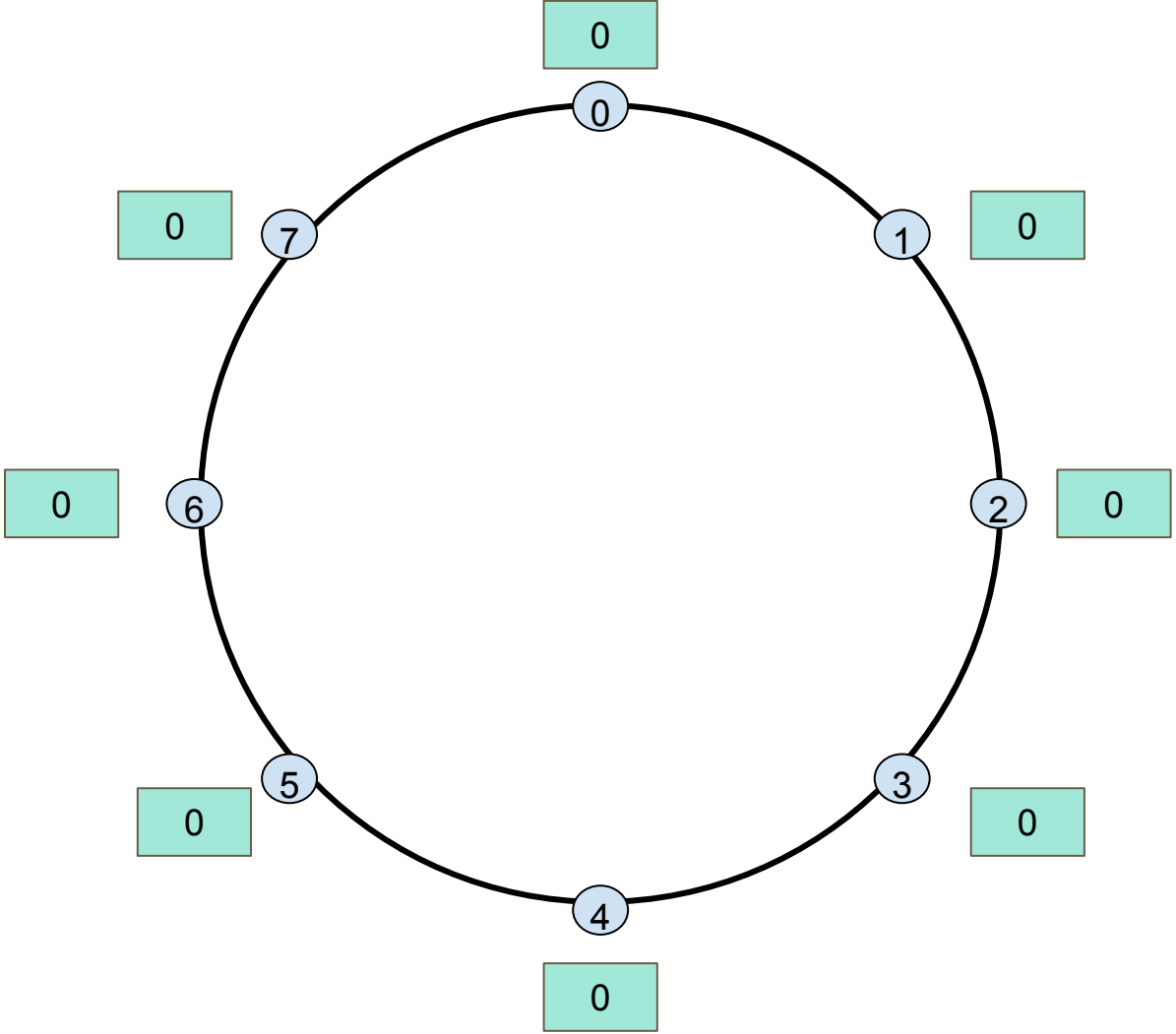
for the Nth machine:

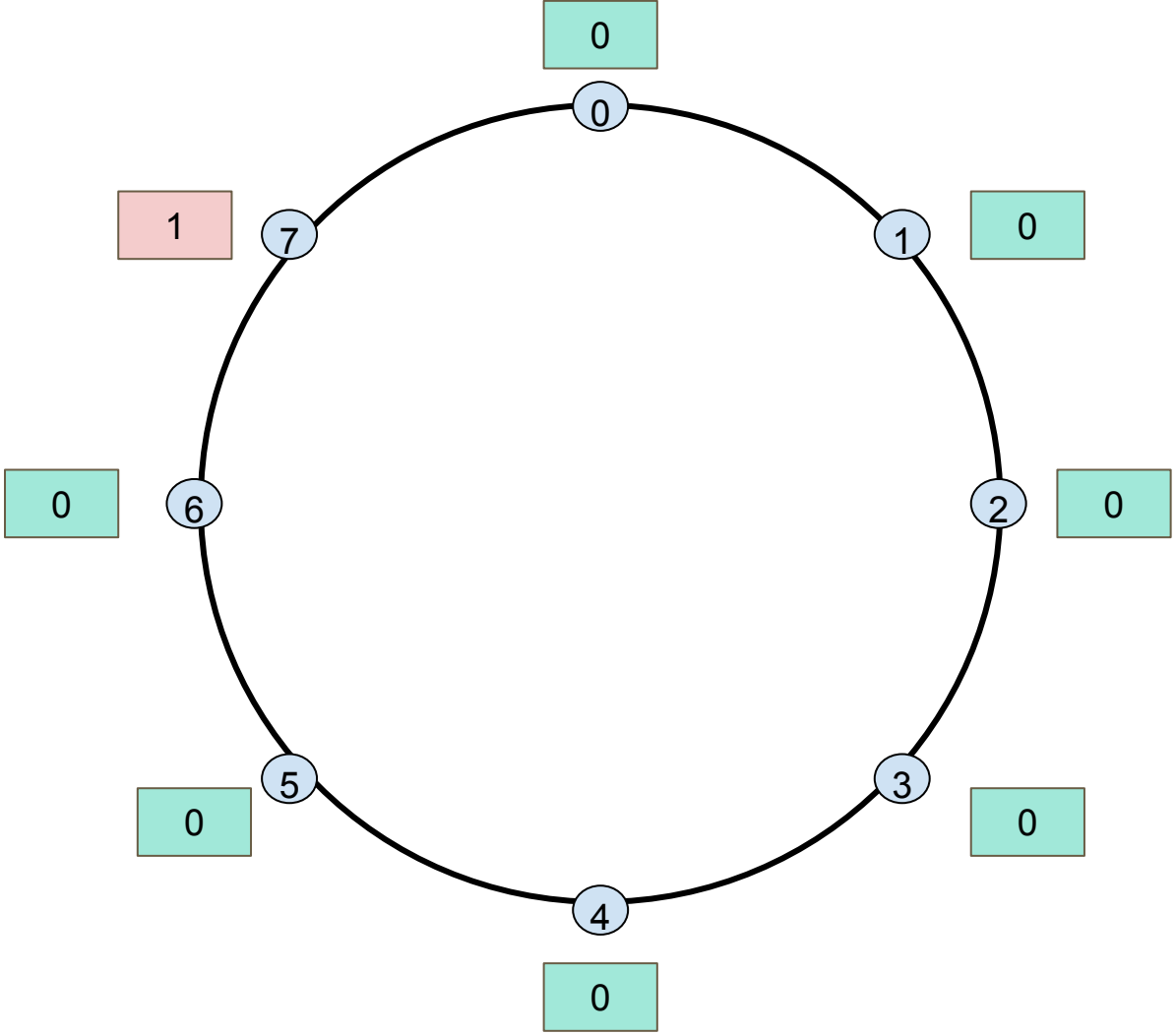
If L = R and (L + 1) mod 3 != S then S := (L + 1) mod 3

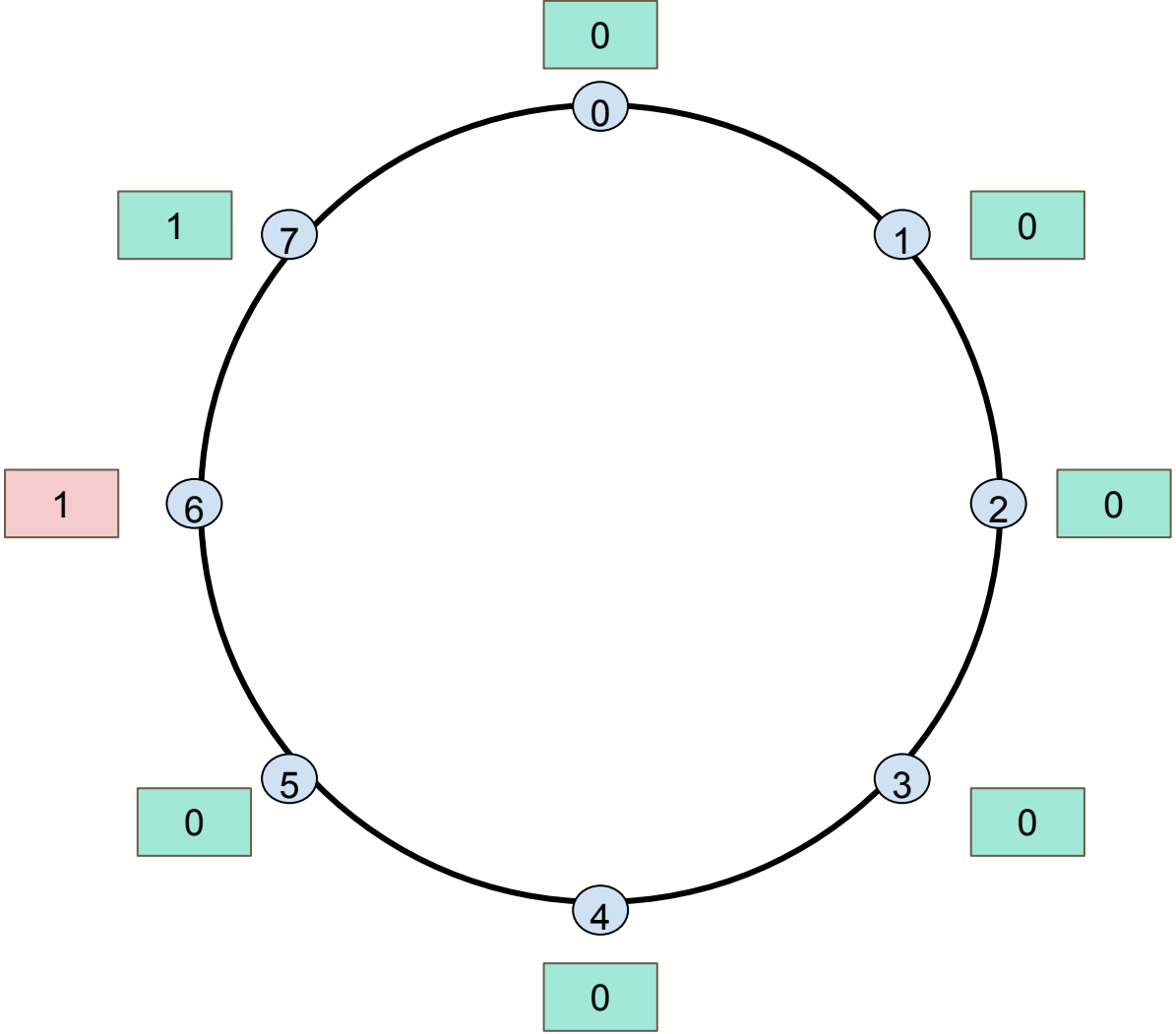
for the other machines:

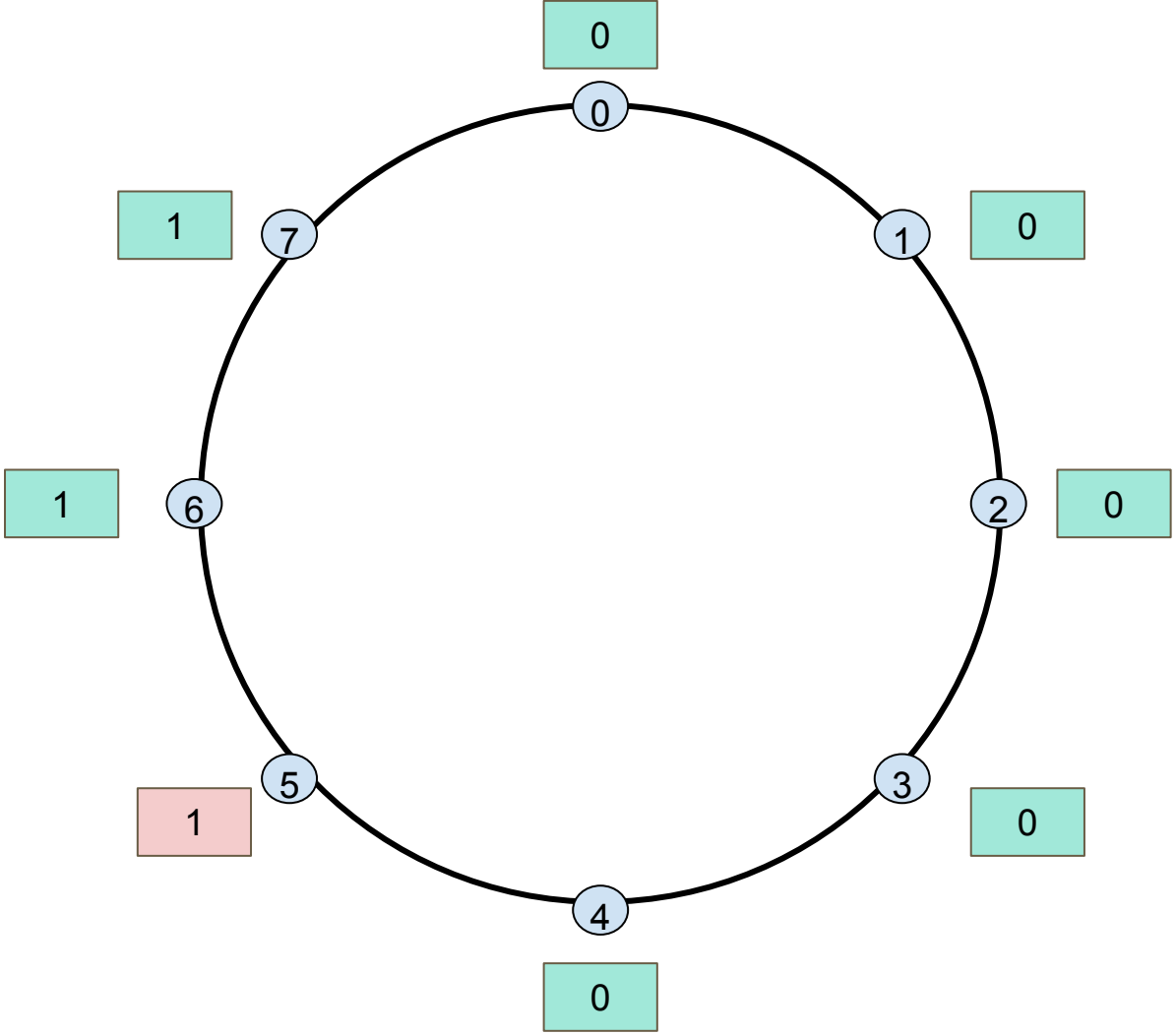
if (S + 1) mod 3 = L then S := L

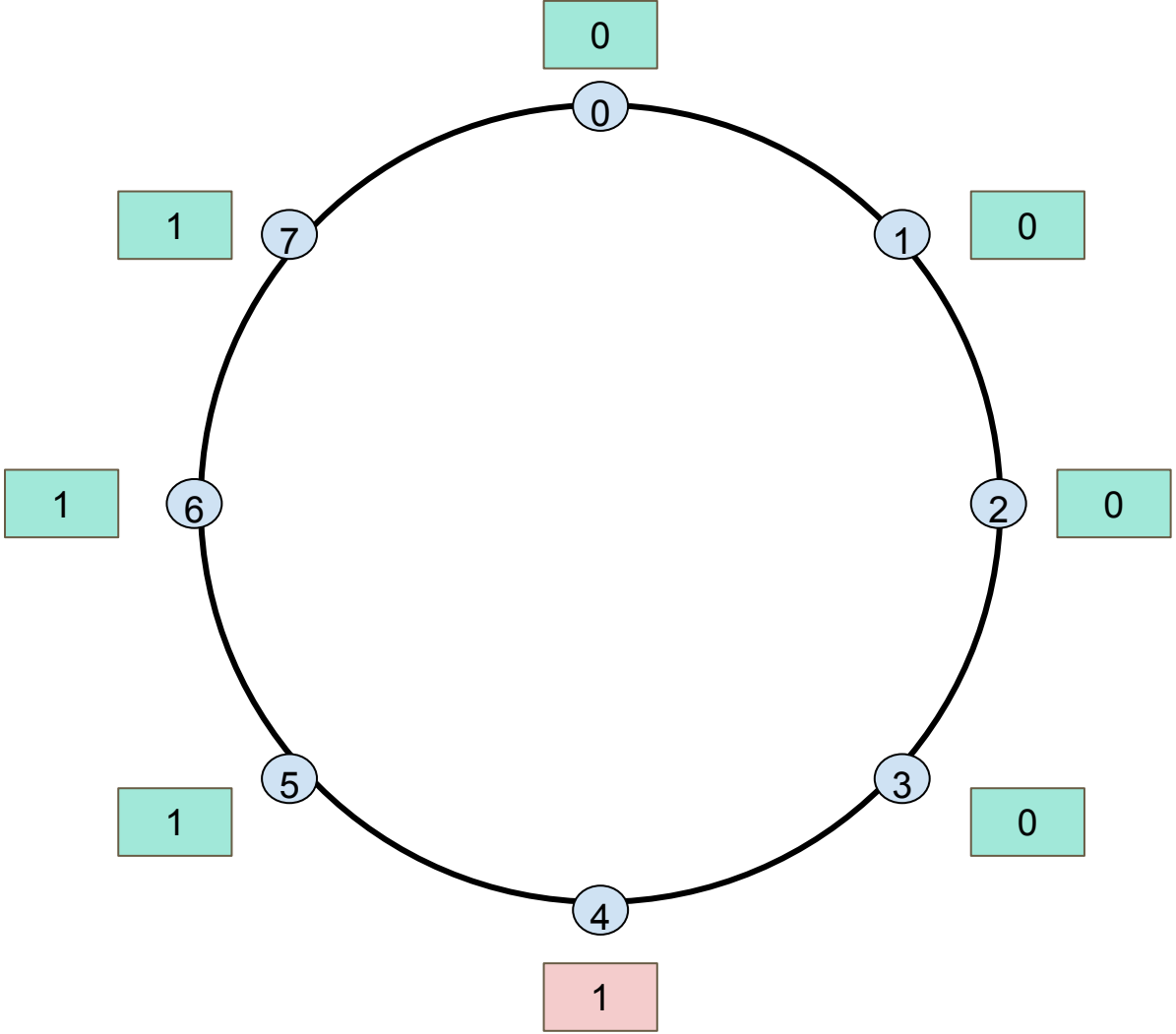
if (S + 1) mod 3 = R then S := R

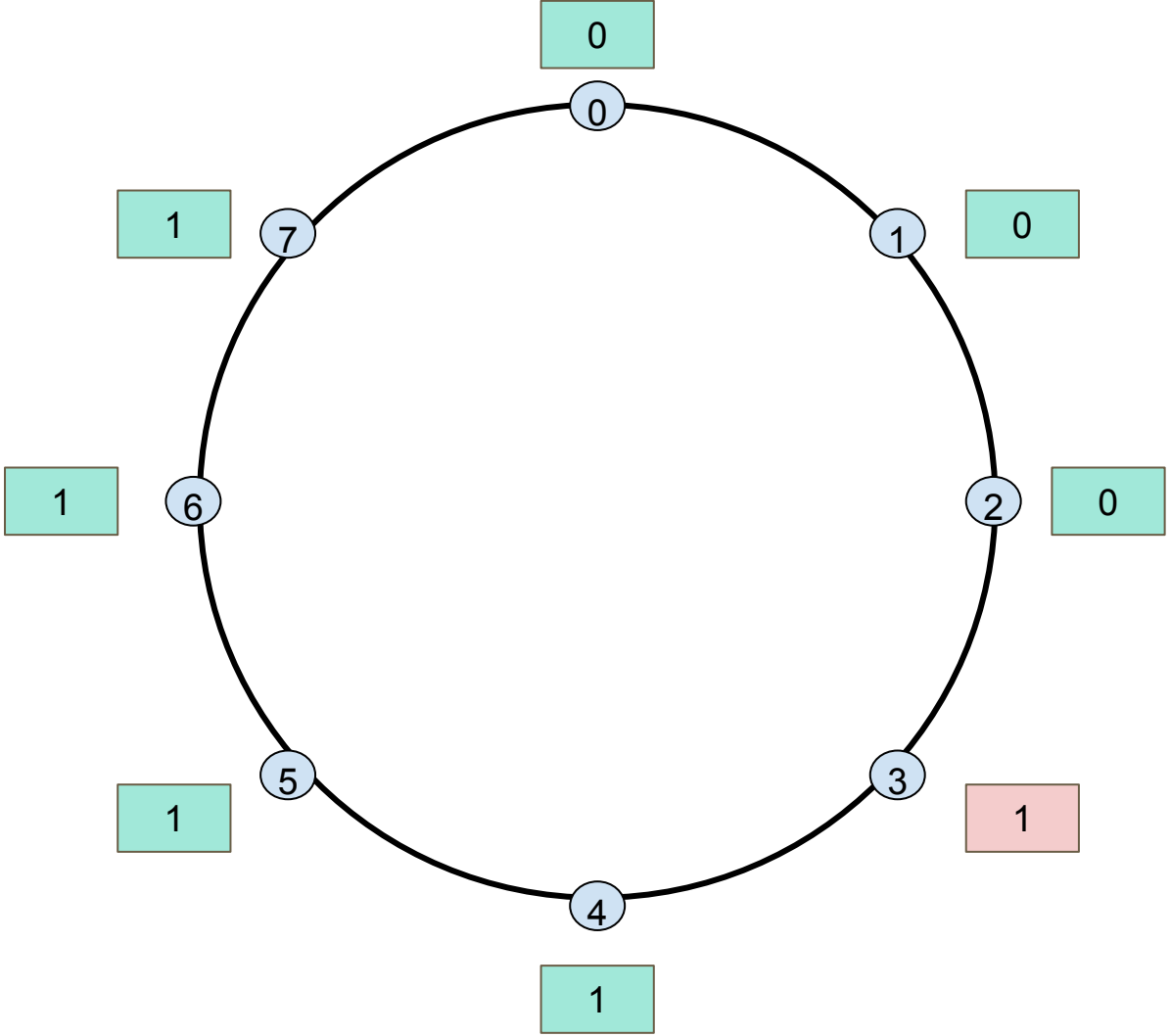

0
0
0
1
0
2
0
3
0
4
0
5
0
6
0
7

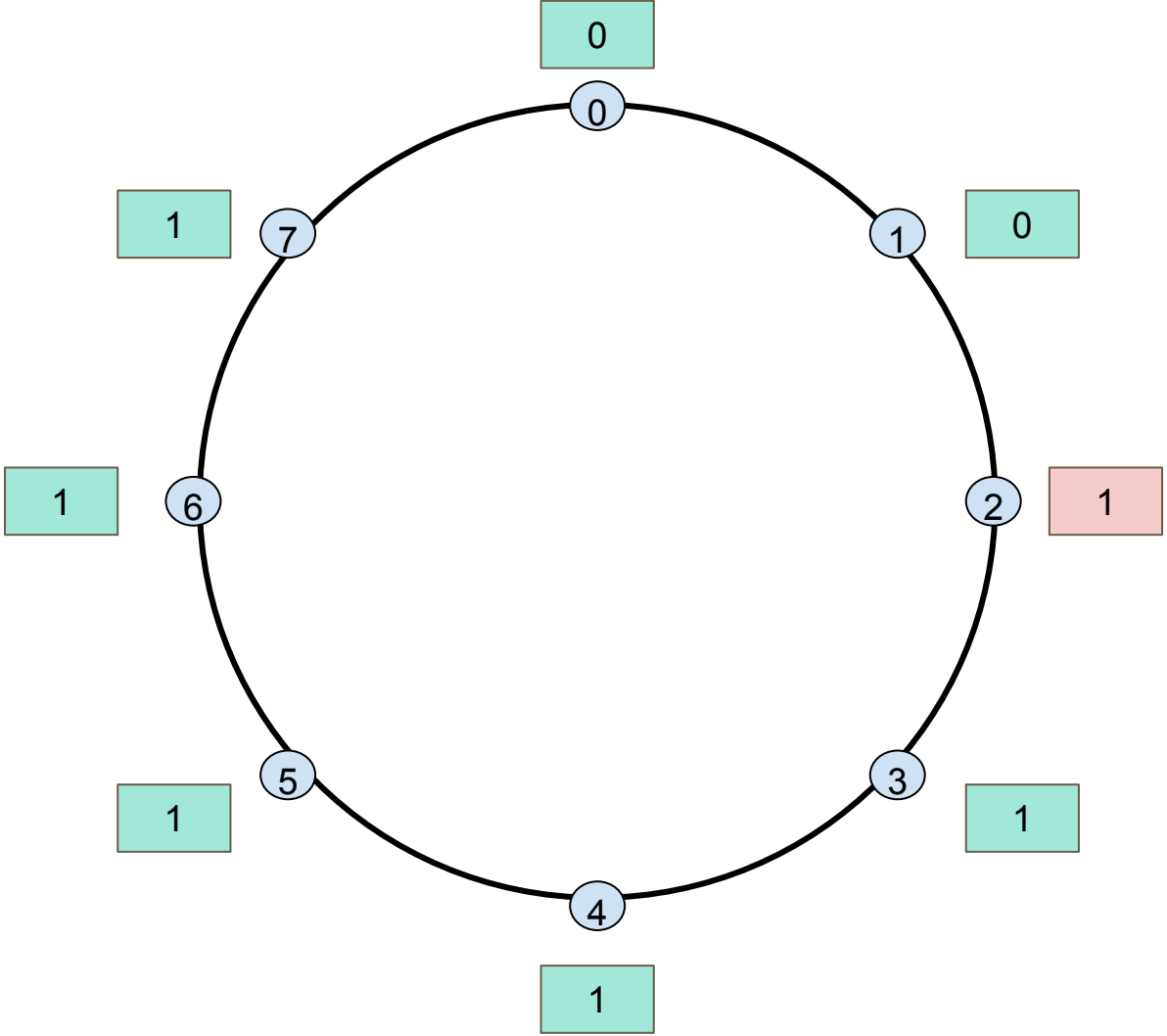

0
0
1
0
7
1
0
6
2
0
5
3
0
4
0

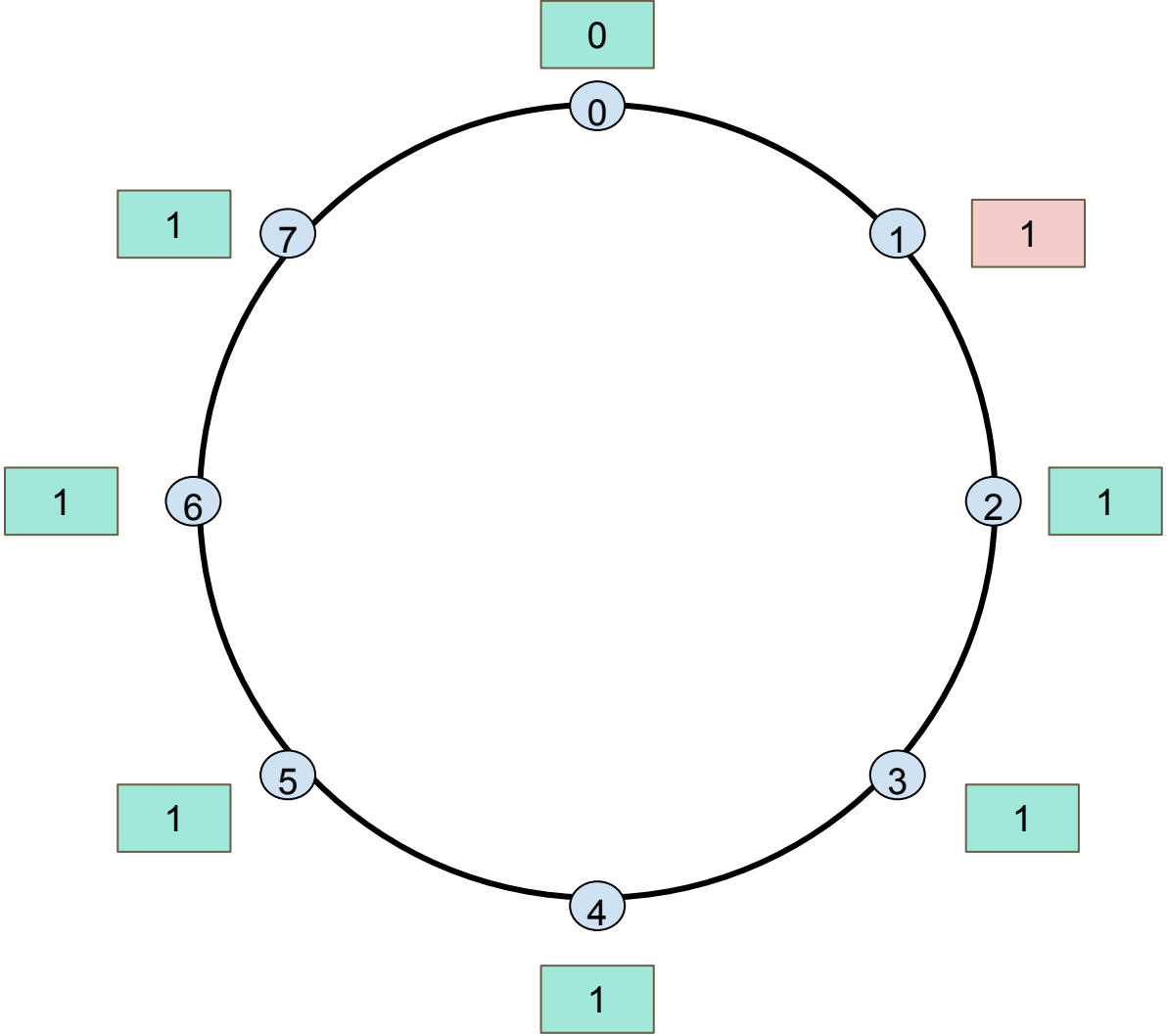

0
0
1
0
2
0
3
0
4
0
5
0
6
1
7
1

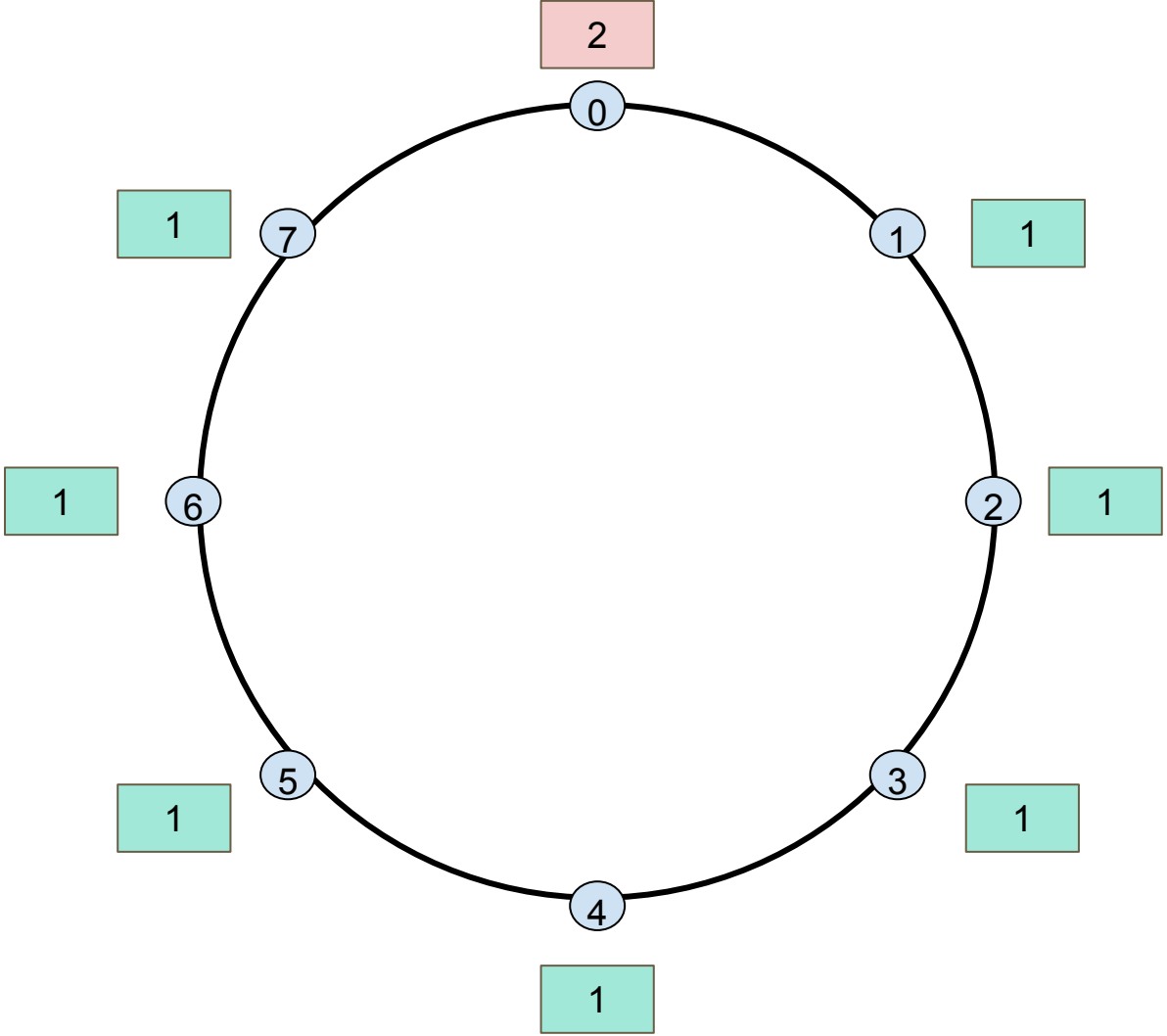

0
0
1
0
2
0
3
0
4
0
5
1
6
1
7
1

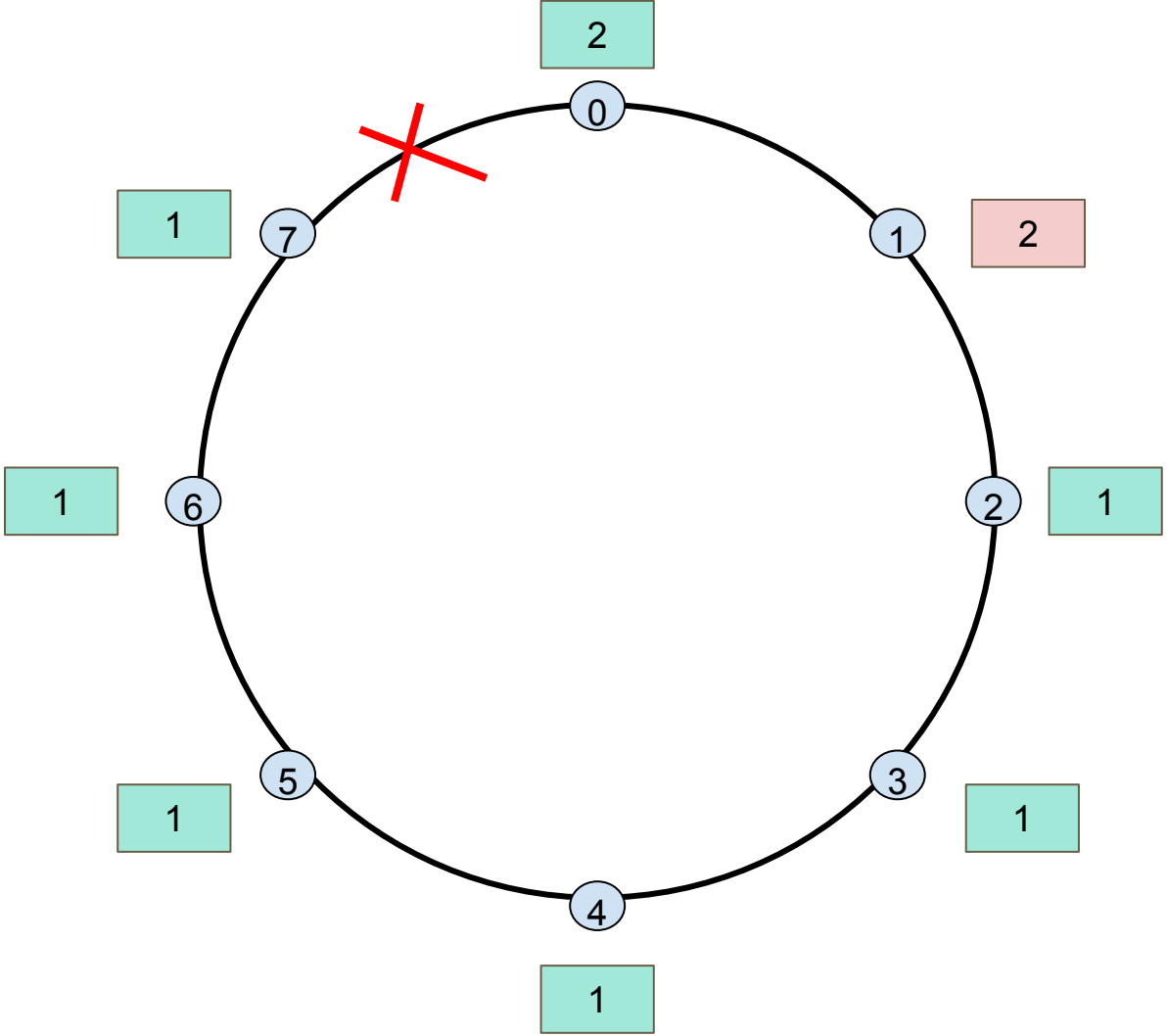

0
0
1
0
2
0
3
0
4
1
5
1
6
1
7
1


0
0
1
0
1
7
0
2
0
1
6
3
1
5
1
4
1


0
0
1
0
7
1
1
6
2
1
5
3
1
1
4
1


0
0
1
1
1
7
1
2
1
6
1
3
1
5
1
4
1


2
0
1
1
1
7
1
6
1
2
1
5
1
3
1
4
1


2
0
1
2
1
2
1
1
6
2
1
1
5
3
1
1
4
1

Drawback:

A central daemon is required

Due to undefined speed ratios of different machines

Only concerned with ring systems - Left and Right neighbors

Impact:

1978 Kruijer, H. S. M.

Self-stabilization (in spite of distributed control) in tree structured systems

1990 Herman, T.

Probabilistic self-stabilization for a unidirectional communication ring with identical processes.

1990 Katz, S., and Perry, K. J.

Self-stabilizing extensions for message-passing systems, contrasted the difficulties of self-stabilization in this model with those of the more common shared-memory models

Current Research:

Towards Self-Stabilizing Operating Systems -- Shlomi Dolev & Reuven Yagel

Self-Stabilizing Distributed File Systems

Self-stabilizing protocols for sensor networks -- D.Bein & A.K.Datta

.... & more

Thank you