MUTUAL EXCLUSION WITHIN RING TOPOLOGY (1)

Simple method:

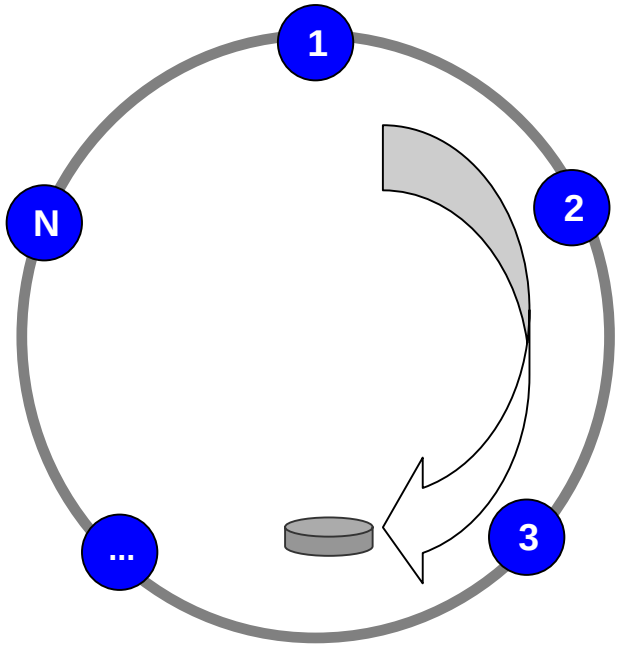
- token passing

Failure model:

- transient link failures

Possible defects:

- token loss



MUTUAL EXCLUSION WITHIN RING TOPOLOGY (2)

Ping-Pong algorithm (Misra 1983)

System model:

- $\boldsymbol{P}_0 \ldots \boldsymbol{P}_{N-1} \rightarrow$ N unknown
- knowledge of P_i's neighbor identities not required
- uniform algorithm (no distinguished process)
- unreliable FIFO channels

Two tokens:

- *ping* – circulating **privilege**, granting entry to the critical section
- *pong* – circulating after ping
- aim of each is to detect a loss of the other

Assumption:

- only one token gets lost in one round

MUTUAL EXCLUSION WITHIN RING TOPOLOGY (3)

Initialization:

- #ping := +1
- #pong := -1
- m_i := 0

  m_i is used by $\boldsymbol{P}_i$ to remember the last token passed trough

Invariant:

- #ping + #pong = 0

MUTUAL EXCLUSION WITHIN RING TOPOLOGY (4)

INSTYTUT INFORMATYKI

P_i:

```
when received PING(#ping)  do
    if m_i == #ping then    // pong is lost, regenerate it
        regenerate()

when received PONG(#pong)  do
    . . .       // similar to above

when meeting PING and PONG  do   // new incarnation of tokens
    incarnate()

when sending PING(#ping) forward  do
    m_i := #ping

when sending PONG(#pong)  do
    . . .       // similar to above
```

MUTUAL EXCLUSION WITHIN RING TOPOLOGY (5)

INSTYTUT INFORMATYKI

P_i:

```
when received PING(#ping)  do
    if m_i == #ping then    // pong is lost, regenerate it
        regenerate(#ping)

when received PONG(#pong)  do
    . . .       // similar to above

when meeting PING and PONG  do   // new incarnation of tokens
    incarnate(#ping)

when sending PING(#ping) forward  do
    m_i := #ping

when sending PONG(#pong)  do
    . . .       // similar to above
```

MUTUAL EXCLUSION WITHIN RING TOPOLOGY (6)

P_i:

```
. . .

regenerate(x)  // restore lost PING or (and) PONG
    {
        #ping := |x|
        #pong := -#ping
    }

incarnate(x)   // increment absolute value of PING and PONG
    {
        #ping := |x| + 1
        #pong := -#ping
    }
```

MUTUAL EXCLUSION WITHIN RING TOPOLOGY (7)

Values of #ping, #pong

- values can be bounded (e.g. for comparisons)
- their absolute values may increase at each P_i
- it is sufficient to increment them modulo N+1

Channel requirements:

- we need FIFO channels
- what would happen if tokens could overpass each other in channel ...?

- ... it will be considered as a loss of the other token – regeneration
- effect : 2 PINGs (or PONGs)
- can we detect such inappropriate regeneration and delete unnecessary token ?
- which one ?

MUTUAL EXCLUSION WITHIN RING TOPOLOGY (8)

Old token deletion:

```
when received PING(#ping) do
    if |#ping| < |m_i| then
                // some old PING has arrived – delete it
    else if m_i == #ping then
        {
         . . .
```

(be aware that it can happen that somewhere in the ring the old ping comes before the new ping)

Problems:

- What about the mutual exclusion in this case?
- What about bounding modulo N+1?
- How can we protect the system from loosing 2 tokens?

BIBLIOGRAPHY

[1] Jayadev Misra, “Detecting Termination of Distributed Computations Using Markers” Proceedings of the 2nd annual ACM symposium on Principles of Distributed Computing (PODC), Montreal, Canada, August 17-19, 1983.

[2] Michel Raynal, "Distributed Algorithms and Protocols", John Wiley & Sons, 1988.