Distributed Computing
Xidian 2021

Prof. Mirosław Kutyłowski

7: Consensus

Consensus problem:

- number of nodes is n, at most f of them faulty
- each node gets an input value $\in \{0, 1\}$
- finally, the correct nodes decide upon a common value
- the common value must be one of the input values

0 0 0 0 0 0 0 $\longrightarrow$ 0

0 1 0 1 0 $\longrightarrow$?

Impossibility of deterministic consensus in asynchronous model with failures

- arbitrary delays but not longer than 1 time unit

- each node executes a deterministic algorithm

- for simplicity – binary inputs only

Configuration

1) States of all nodes

2) Messages in transit

univalent configuration: decision value can be only one no matter what happens later

b-configuration: univalent, decision will be b

bivalent: decision will be 0 or 1

Lemma

Let f>1.

There is at least one bivalent initial configuration

e.g. consider inputs (n=8):

00000000 ← 0-conf

10000000 ← 0-conf

2 11000000 ← bivalent

11100000 ← 1-conf

11110000 ← ~~0-conf~~

11111000 ← ~~1-conf~~

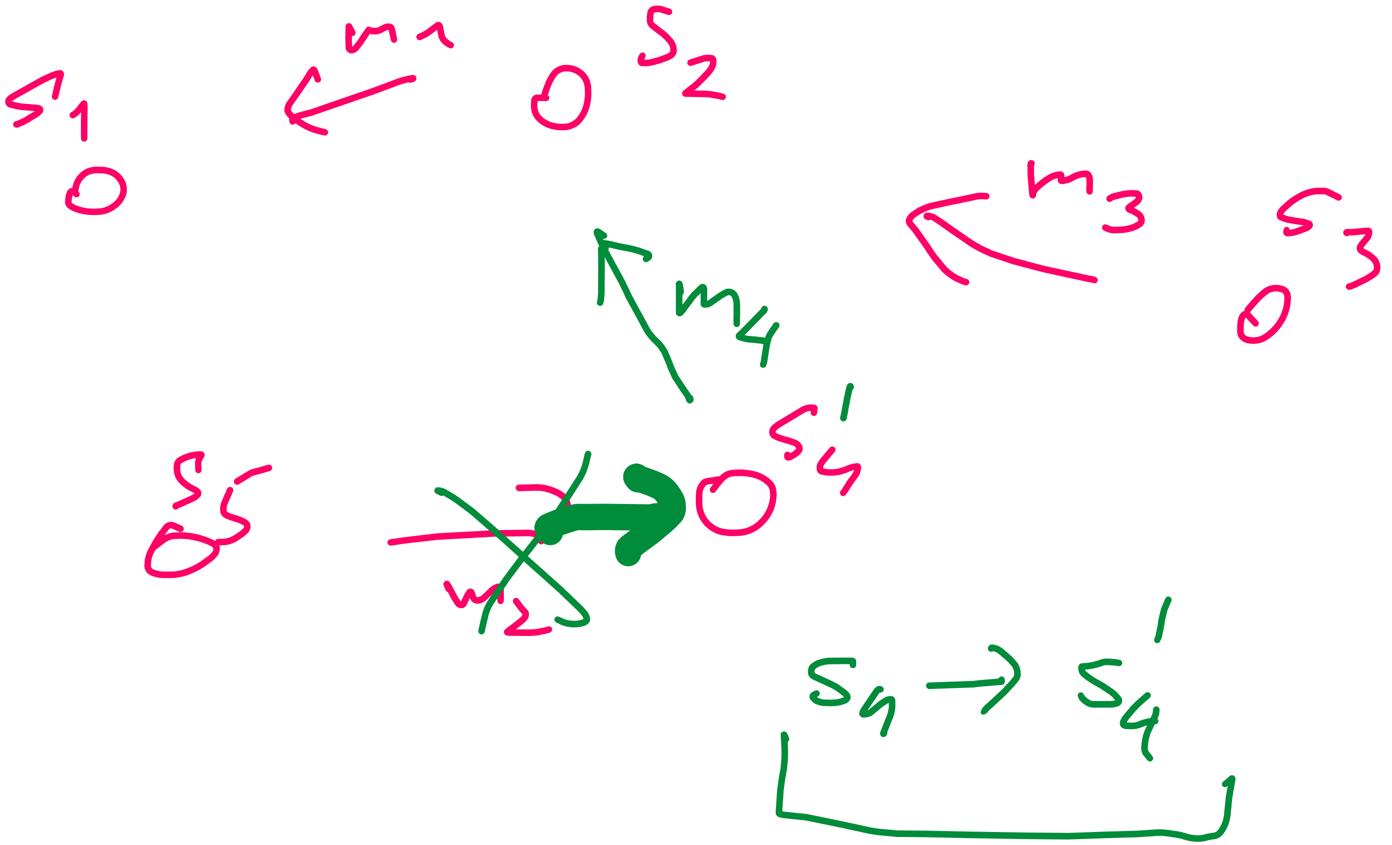
11111100

11111110

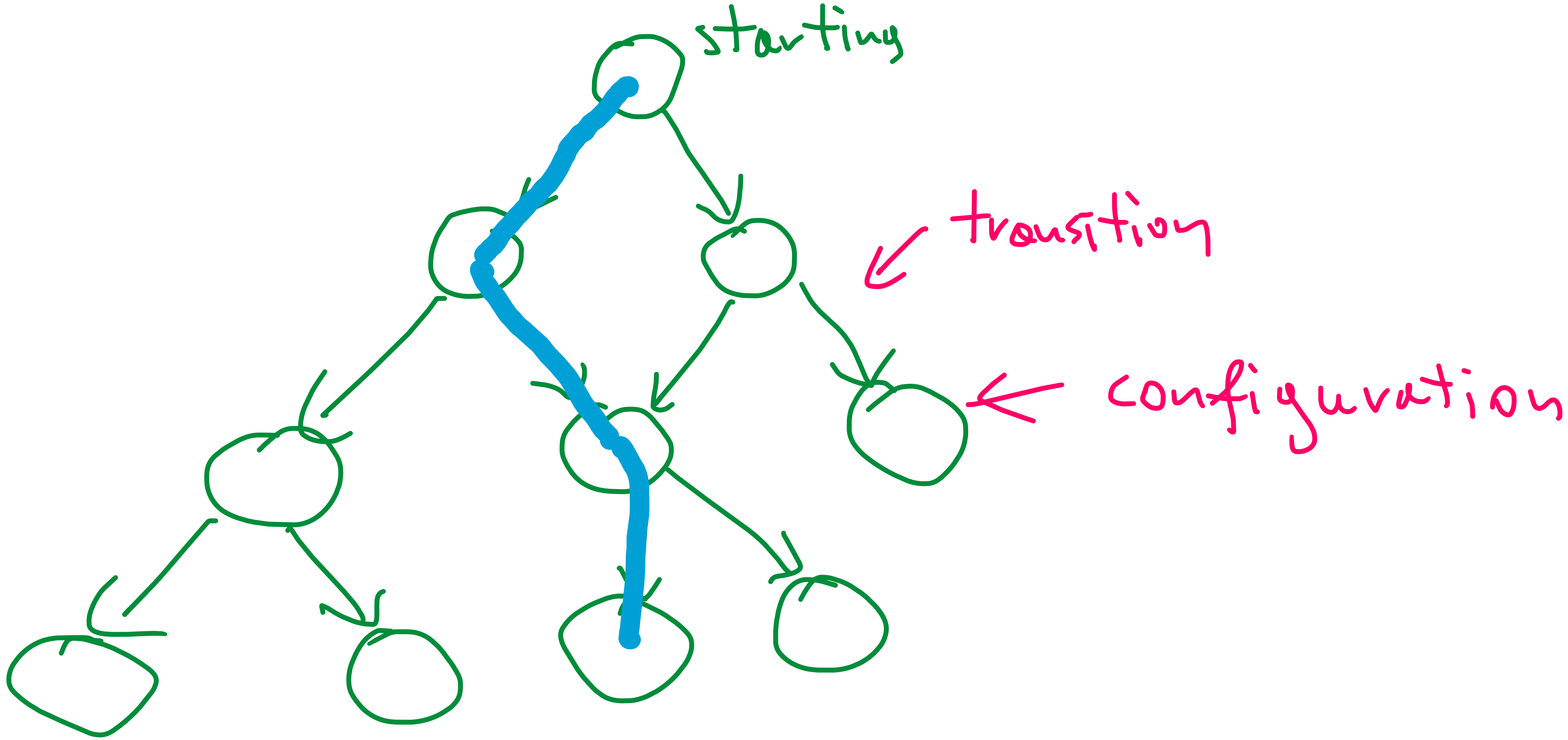
11111111 ← 1-conf

Configuration transition

- One message delivered, state changed, new messages despatched



Configuration directed graph

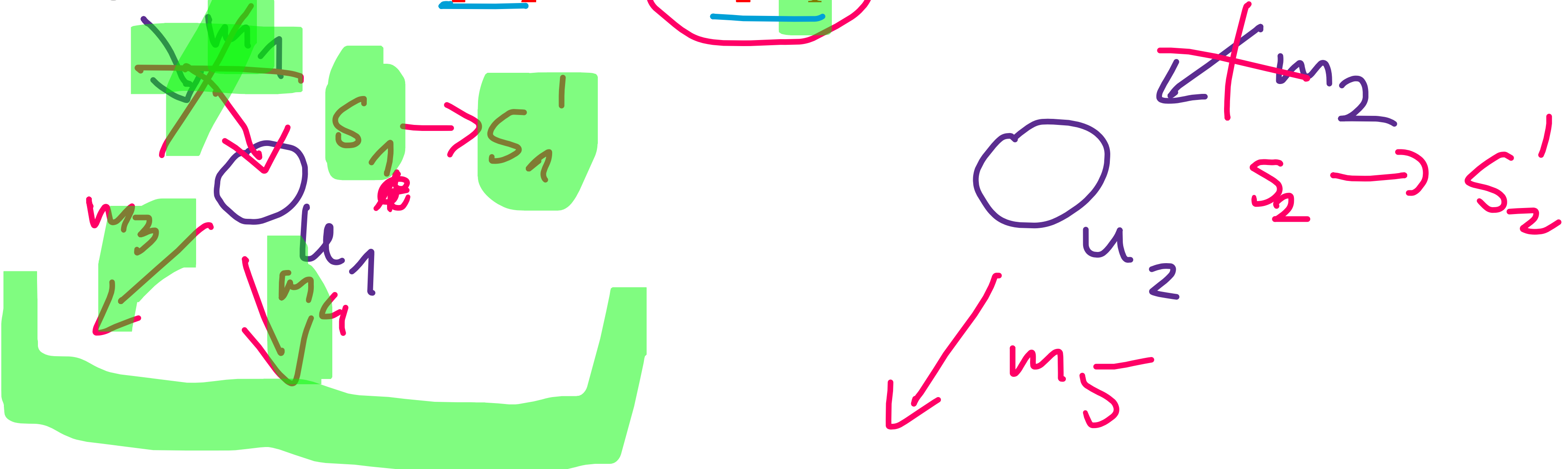


Two transitions

- in a state S two transitions are possible $\tau_1=(u_1,m_1)$ and $\tau_2=(u_2,m_2)$,
- where $u_1 \neq u_2$

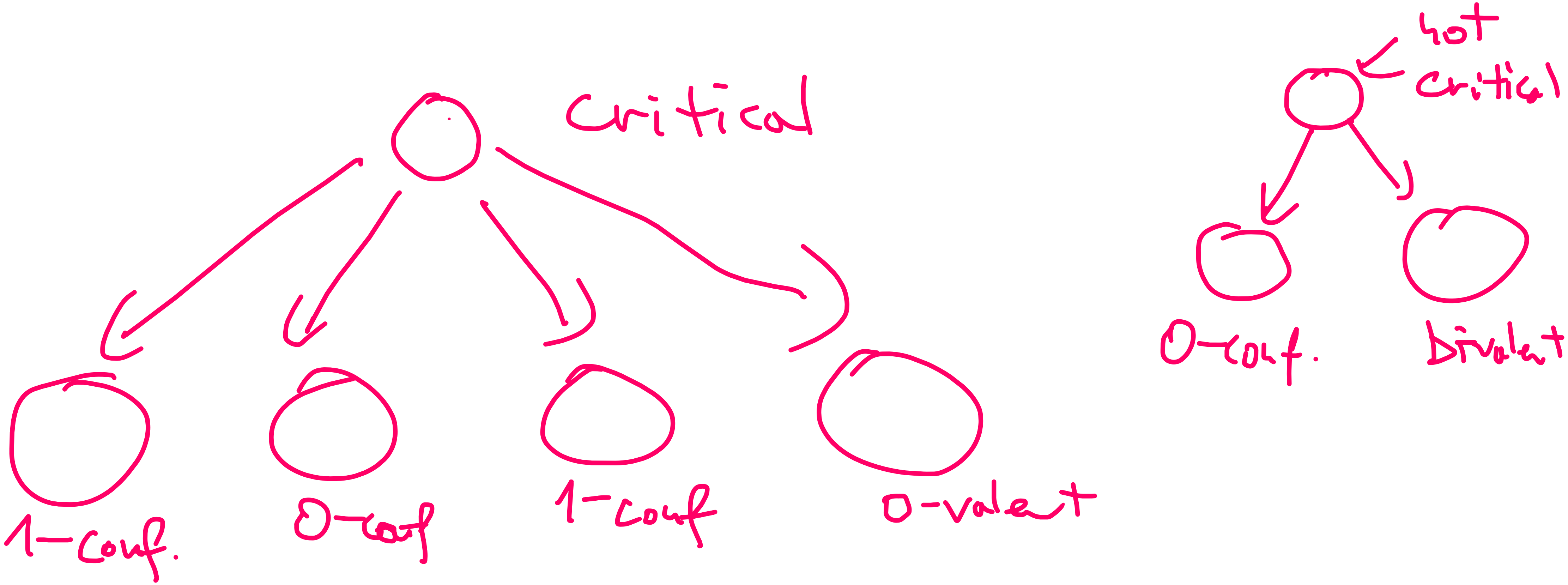
then

configurations $S(\tau_1, \tau_2)$ and $S(\tau_2, \tau_1)$ are the same

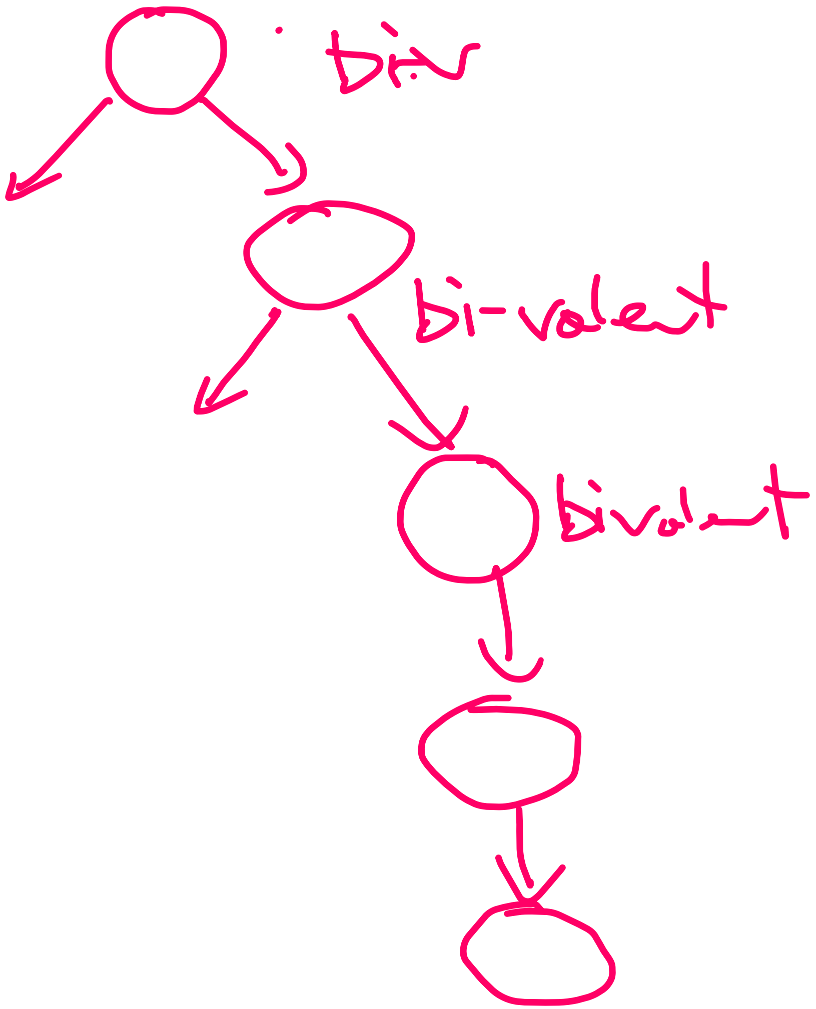


Critical configuration

A bivalent configuration such that each child configuration is univalent

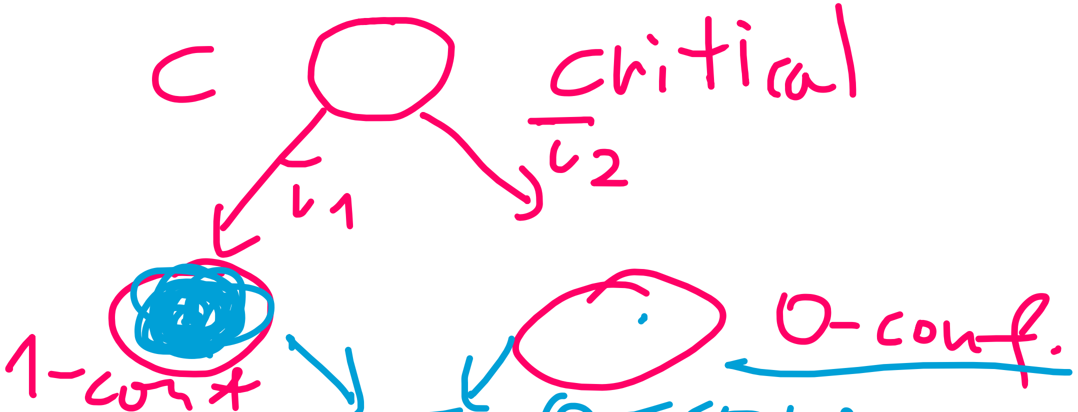


Lemma 16.12. *If a system is in a bivalent configuration, it must reach a critical configuration within finite time, or it does not always solve consensus.*



Lemma 16.13. *If a configuration tree contains a critical configuration, crashing a single node can create a bivalent leaf; i.e., a crash prevents the algorithm from reaching agreement.*

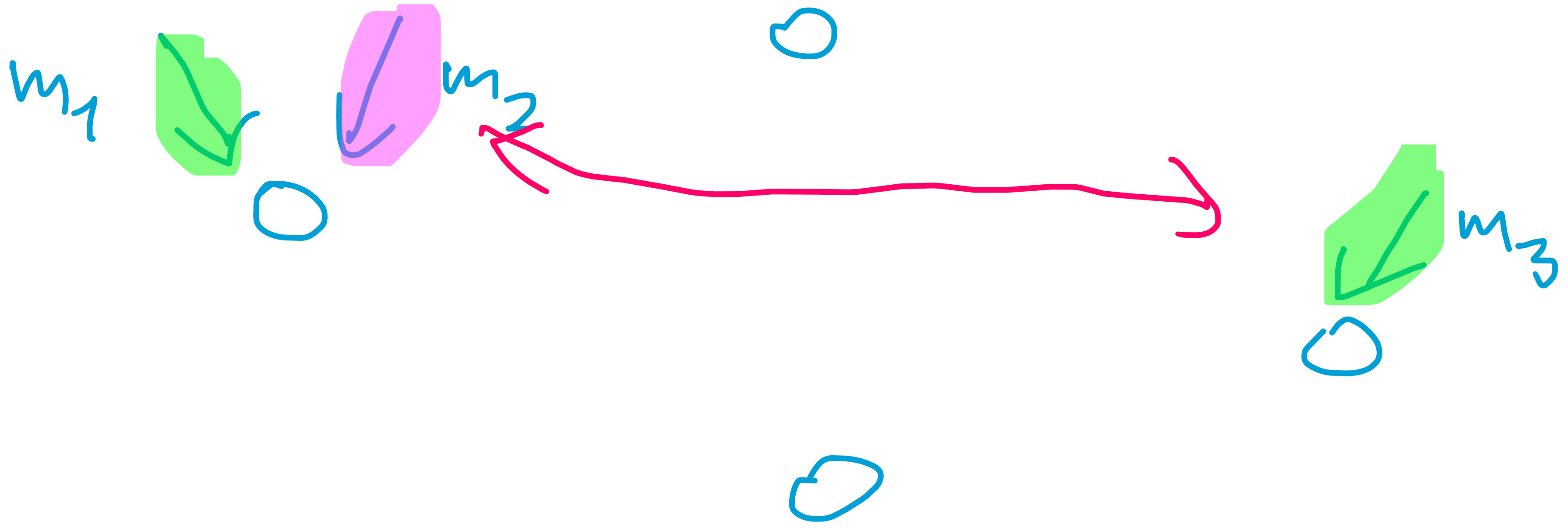
- Let C be a critical configuration
- Let $\tau_1=(u_1,m_1)$ and $\tau_2=(u_2,m_2)$ be transitions such that
  - $C(\tau_1)$ is 1-valent
  - $C(\tau_2)$ is 0-valent
- If $u_1 \neq u_2$ then $C(\tau_1\ \tau_2)=C(\tau_2\ \tau_1)$, so it is 1-valent and 0-valent at the same time !?
- Consequently: $u_1=u_2$



Corollary

Crash it

All transitions from a critical configuration C must involve the same node



FINAL OBSERVATION

crush this node at configuration C
No consensus can be reached

Theorem 16.14. *There is no deterministic algorithm which always achieves consensus in the asynchronous model, with* $f > 0$*.*

Randomized consensus

Deterministic solution is impossible, but …

… what about a randomized algorithm with random execution time (and always correct output)? (Las Vegas algorithm)

Monte Carlo – fixed time
results sometimes are wrong

Algorithm 16.15 Randomized Consensus (Ben-Or)

$v_i \in \{0, 1\}$ ◁ input bit
round = 1
decided = false

Broadcast `myValue`(v_i, round)

while true **do**

myValue(0, 1)

Propose

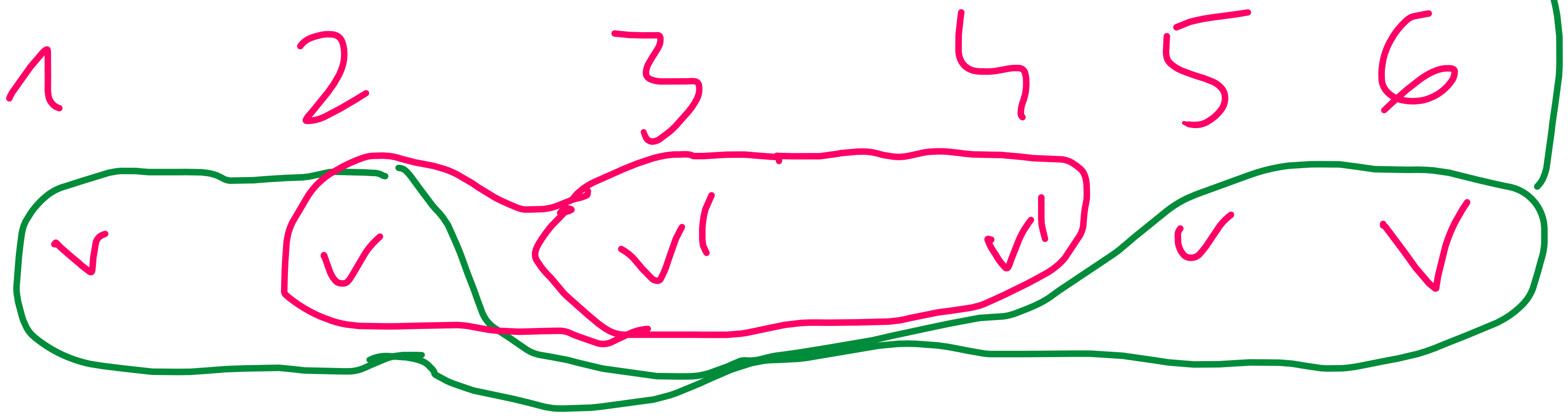
```
Wait until a majority of myValue messages of current round arrived
if all messages contain the same value v then
    Broadcast propose(v, round)
else
    Broadcast propose(⊥, round)
end if

if decided then
    Broadcast myValue(v_i, round+1)
    Decide for v_i and terminate
end if
```

nodes:

Myvalue

proposition:



312 312 read

Adapt

Wait until a majority of **propose** messages of current round arrived
if all messages propose the same value v **then**
    $v_i = v$
    decide = true
else if there is at least one proposal for v **then**
    $v_i = v$
else
    Choose v_i randomly, with $Pr[v_i = 0] = Pr[v_i = 1] = \cancel{1/2}$
end if
round = round + 1
Broadcast **myValue**(v_i, round)
end while

u'

shared coin algo

u - decided, majority with v

u' - another majority, they intersect!

All 0 input

- It finishes quite quickly!

Lemma 16.16. *As long as no node sets* decided *to true, Algorithm 16.15 always makes progress, independent of which nodes crash.*

What happens if all input bits are the same?

```
    Wait until a majority of myValue messages of current round arrived
    if all messages contain the same value v then
        Broadcast propose(v, round)
    else
        Broadcast propose(⊥, round)
    end if
    if decided then
        Broadcast myValue(v_i, round+1)
        Decide for v_i and terminate
    end if
    Adapt
    Wait until a majority of propose messages of current round arrived
    if all messages propose the same value v then
        v_i = v
        decide = true
    else if there is at least one proposal for v then
        v_i = v
    else
        Choose v_i randomly, with Pr[v_i = 0] = Pr[v_i = 1] = 1/2
    end if
    round = round + 1
    Broadcast myValue(v_i, round)
end while
```

Essential case – both 0 and 1 as input

- any consensus value is ok
- no proposals for different bits in the same round
- u – the first node that decides, say: for v, at round r
  1) - no decide at this round to a different v'
  - u terminates at the round r+1

case: Propose(v, r)
Propose(1, r)

Termination

- u – the first node that decides, say: for v, at round r
  - no decide at this round to a different v’
  - u terminates at the round r+1
- An other node u’
  - If decides, then only for the same v
  - otherwise, it has heard at least one propose(v,r) and sets $v_{u'}$ to v
- So all nodes broadcast v at the end of round r

Runtime

quite bad, since the choices in line 23 must be almost the same

⇒ exponential time

It would help to have a public coin and to toss it together!

(it seems to be a hopeless task in a distributed system)

input

0 0 0 0 1 1 1 1

Shared coin

1/8

Random variable that is equal to 0 with probability >1/4 and 1 with probability >1/4

(the value of the shared coin is the same for all nodes)

Algorithm 16.22 Shared Coin (code for node u)

Choose local coin $c_u = 0$ with probability $1/n$, else $c_u = 1$
Broadcast `myCoin`(c_u)
Wait for $n - f$ coins and store them in the local coin set C_u
Broadcast `mySet`(C_u)
Wait for $n - f$ coin sets
if at least one coin is 0 among all coins in the coin sets **then**
    return 0
else
    return 1
end if

We show the correctness of the algorithm for $f < n/3$. To simplify the proof we assume that $n = 3f + 1$, i.e., we assume the worst case.

tossing own coin:

$$\left(1-\frac{1}{n}\right)^n$$

$\approx$

$$\frac{1}{e} > \frac{1}{4}$$

!

$\underbrace{\text{no zero}}$ $\Rightarrow$ no coin set contains 0

$\Rightarrow$ decisions $= \underline{1}$

Lemma 16.23. *Let u be a node, and let W be the set of coins that u received in at least $f+1$ different coin sets. It holds that $|W| \geq f+1$.*

C be the multiset of coins received by u.

$|C| = (n-f)^2$

Assuming that the lemma is false

$|C| \leq f \cdot (n-f) + (n-f) \cdot f = 2f(n-f)$

, $n - f > 2f$

$|C| \leq 2f(n-f) < (n-f)^2 = |C|$

Lemma 16.24. *All coins in W are seen by all correct nodes.*

Theorem 16.25. *If $f < n/3$ nodes crash, Algorithm 16.22 implements a shared coin.*

With probability $(1 - 1/n)^n \approx 1/e \approx 0.37$ all nodes chose their local coin equal to 1 (Line 1), and in that case 1 will be decided.

With probability $1 - (1 - 1/n)^{|W|}$ there is at least one 0 in W

$|W| \geq f + 1 \approx n/3,$

$1 - (1 - 1/n)^{n/3} \approx 1 - (1/e)^{1/3} \approx 0.28$