DS

2º PARTE

MOVEMENT ON GRAPH

ISTANZA

IN $G=(V,E)$ t/c $|V|=n$
P t/c $|P|=k$
$\sigma: P \to V$

OUT $\mu: P \to V$

GOALS sia $U := \{\mu(p)\}_{p \in P}$ t/c $U \subseteq V$:

- CONNECTIVITY sottografo indotto da U connesso
- INDIPENDENCY U indipendente e $|U|=k$
- CLIQUE U clique di G

MEASURES $p \in P$ spostato da $\sigma(p)$ a $\mu(p)$ tramite lo shortest path sul grafo

- OVERALL MOVEMENT
  $SUM(\mu) = \sum_{p \in P} d_G(\sigma(p), \mu(p))$
- MAXIMUM MOVEMENT
  $MAX(\mu) = \max_{p \in P} d_G(\sigma(p), \mu(p))$
- #PEBBLE MOVED
  $NUM(\mu) = |\{p \in P \text{ t/c } \sigma(p) \neq \mu(p)\}|$

IND-MAX HARDNESS

IS $U \subseteq V$ DI G T/C $\forall u,v \in U \; \exists (u,v)$

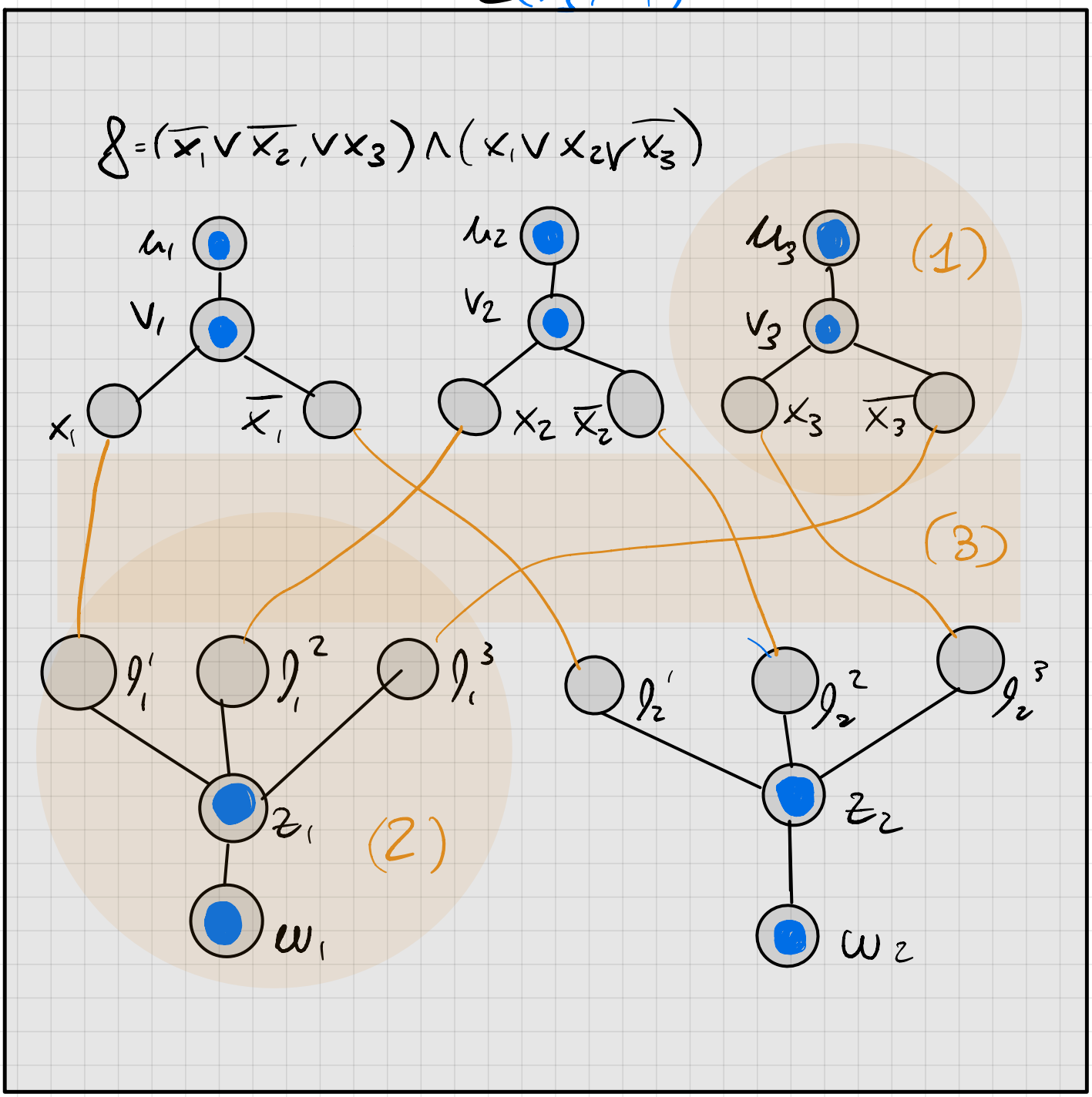
max IS $U^* \subseteq V$ T/C $|U^*| > U \; \forall U$

3 SAT -> IND MAX

(1) VARIABLE GADJET $\forall x_i \in \varphi$

(2) CLAUSE GADJET $\forall c_j = (l_j^1, l_j^2, l_j^3)$

(3) $\forall l_j^i \in c_j$ INTRODUCI $\begin{cases} (l_j^i, x_i) \text{ SE } l_j^i = \overline{x}_i \\ (l_j^i, \overline{x}_i) \end{cases}$



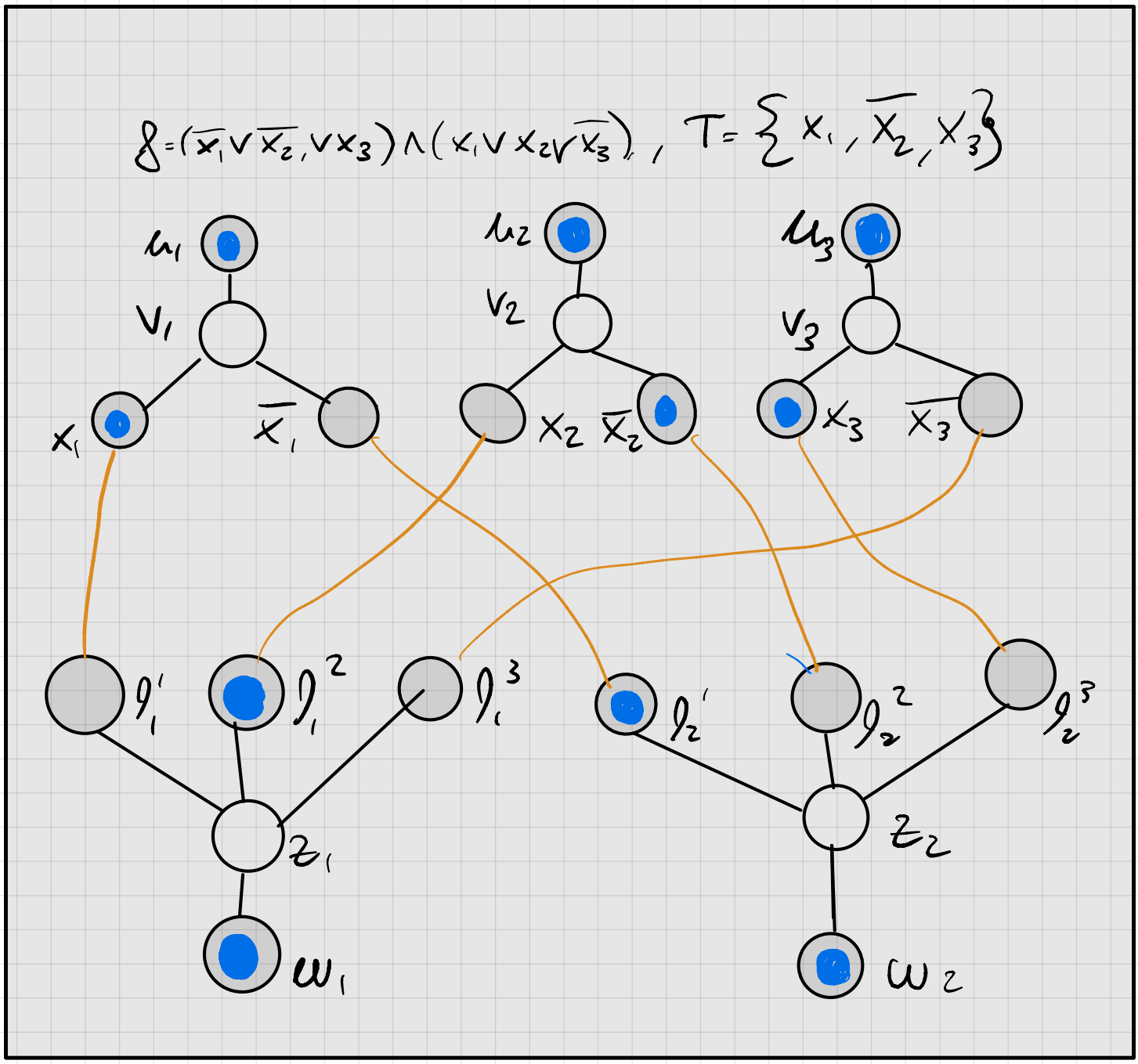
CLAIM φ SODDISFACIBILE $\Leftrightarrow$ $\exists$ SOL PER IND-MAX DI COSTO 1

Proof

($\Rightarrow$)
- CONSIDERO T PER φ
- $\forall x_i \in T$ VERA : SPOSTO P DA v_i A x_i, IN $\bar{x}_i$ ALTRIMENTI
- $\forall q_j^k$ VERO T/C $Adj(q_j^k)$ LIBERO $\Rightarrow$ SPOSTO P DA z_i A q_j^k
- $\forall c_j$ ALMENO UN q_j^k VERO T/C $Adj(q_j^k)$ VUOTO

($\Leftarrow$)
- CONSIDERO U SOLUZIONE IND-MAX
- OGNI P IN V DEVE ESSERE STATO SPOSTATO IN x_i O $\bar{x}_i$, SETTIAMOLO TRUE IN T.
- OGNI P SU z_i DEVE ESSERE STATO SPOSTATO IN q_j^i E $Adj(q_j^i)$ VUOTO
- q_j^i SODDISFATTO $\Rightarrow$ c_v SODDISFATTO

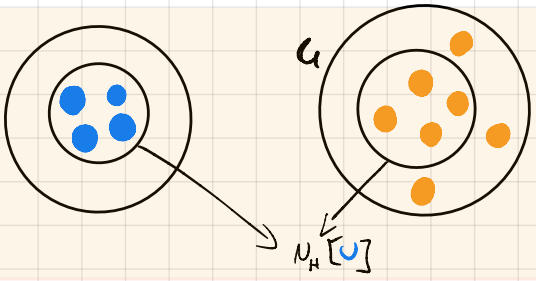
□



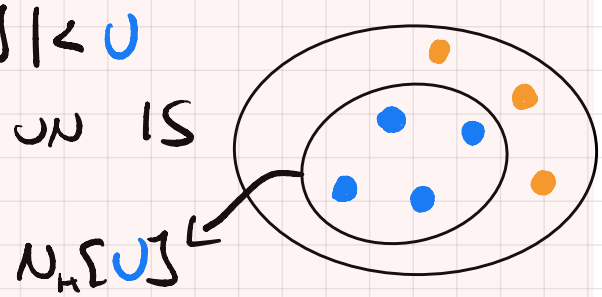
IND-MAX APPROXIMABILITY

TH (HALL'S MATCHING) SIA $H = (V_1 + V_2, E)$ BIPARTITO
$\Rightarrow \exists$ MATCHING M T/C $|M| = |V_1| \iff |A| \leq N_H(A)\ \forall A \subseteq V_1$

LEMMA SIA U^* max IS DI G
$\Rightarrow \forall U$ IS DI $G \Rightarrow$
$|U^* \cap N_H[U]| \geq U_G$

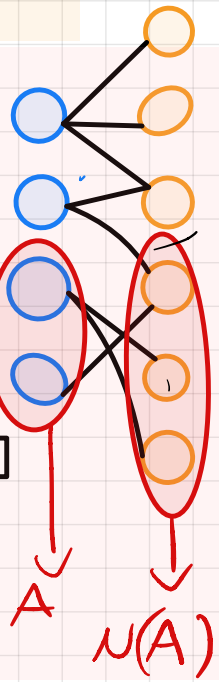


Proof SUPPONIAMO $|U^* \cap N_H[U]| < U$
$\Rightarrow U' = (U^* \setminus N_H[U]) \cup U$ È UN IS
$\Rightarrow |U'| > |U^*| \Rightarrow \perp$ □



LEMMA $\forall$ IS U DI G $\exists f: U \to U^*$ INIETTIVA T/C
$d_G(u, f(u)) \leq 1$

Proof
- COSTRUIAMO G BIPARTITO $H = (U + U^*, E)$
- CONNETTIAMO $u \in U$ A $U^* \cap N[\{u\}]$
- DA HALL'S MATCHING
$\forall A \subseteq U \Rightarrow N(A) = |U^* \cap N_H[A]| \geq A$ □



TH SE UN max IS DI G TROVABILE IN POLY. T (AD ES SU GRAFI BIPARTITI) => IND-MAX TROVABILE CON 1-ADDITIVE ERROR

ALGO TROVA UN IND-MAX DATO U^*

$U^* \leftarrow$ max IS di G

if ($|U^*| < |P|$):
  return NO_SOL

for K=0 TO |V|-1:
  $F = \{(p, u) \in P \times U^* \text{ t.c. } d(\sigma(p), u) \leq k\}$
  $H = (P + U^*, E)$
  M = max_bipartite_matching (H) // HALL'S TH
  if ($|M| = |P|$):
    return M

- L'ADDITIVE ERR È DATO DA:
  - SE HA UN IS È A SPOSTI I PEBBLE CON SOLUZIONE DI COST
  - CON UN ALTRO PASSO (COST+1) HA SPOSTI SU IL MAX IS

DISTRIBUTED VERTEX COLORING

VERTEX COLORING $G(V,E)$ T/C $v \in V$ COLORATO

VALID VERTEX COLORING $\exists\, v_1, v_2 \in V$ ADJ. T/C $c_{v_1} \neq c_{v_2}$

K-COLORING VERTEX COLORING DI K COLS

CHROMATIC NUMBER $\varphi(G) = \min k$ T/C $\exists$ K-COLORING(G)

$\Delta+1$ COLORING

$\Delta = \max_{v \in V} \delta(v) = \delta(G)$

CLAIM OGNI G AMMETTE $\Delta+1$ COLORING, CIOÉ

$$\varphi(G) \leq \Delta + 1 \quad \forall G$$

PROOF

- $\forall v \in V \;\exists$ Palette(v) CON $\delta(v)+1$ COLS DISTINTI
- $r = \#N_{col}(v)$
- MARCA NON DISPONIBILE c_u T/C $u \in N_{col}(v)$
- $\#c_v$-RIMASTI $\geq \delta(v)+1-r \geq \delta(v)+1-\delta(v) = 1$

↳ RIMANE SEMPRE UN COL DISPONIBILE PER v □

ALGO

```
∀v ∈ V creo Palette(v) di δ(v)+1 cols
WHILE ∃ v ∈ V NON COLORATO
    colloro v con c_v ∈ Palette(v) t/c c_v DISP
    FOR u ∈ N(v)
        MARCO c_u ∈ Palette(v) NON-DISP
```

MIS-COLORING

COLORING e IS IN OGNI VALIDO VERTEX COLORING(G), I NODI CON STESSO COLORE FORMANO UN IS

ALGO

```
C = 1
WHILE ∃v∈V NON COLORATO
    TROVA MIS I DEL SOTTOGRAFO INDOTTO DAI v∈V NON COLORATI.
    ASSEGNA C AD OGNI v∈I
    C++
```

ANALYSIS

LEMMA MASSIMO $\Delta+1$ ITERAZIONI

Proof

ITERAZIONE k

GRADO EFFETTIVO $eff\_d(v) = |N_{NON\_COL}(v)|$

END of k

OGNI $v \in V$ ADJ. A $u \in I$ (ALTR. $|I \cup \{v\}| > |I|$ ↯)

$eff\_d(v)$ --

END of $k = \Delta$

$eff\_d(v) = \emptyset$

END of $k = \Delta + 1$

OGNI v RIMASTO ENTRA IN I □

COMPLEXITY

$\underbrace{O(\Delta+1)}_{\#ITERATIONS}\ \underbrace{O(\log\Delta \log n)}_{LUBY\ -\ HIGH\ PROB.} = O(\Delta \log\Delta \log n)$ HIGH PROB.

2 Δ-COLORING

$|Palette(v)| = 2\delta(v)$

SE v NON COLORATO ALLA FASE k
- SCEGLI $c_v \in Palette(v)$ CANDIDATO
- SE $c_v \neq c_u \ \forall u \in N(v)$ DEFINITIVO
- SENNO RIGETTA c_v

I NODI CHE RIGETTANO PASSANO A k+1

TERMINA QUANDO OGNI v COLORATO

ALGO (Per $v \in V$)
- WHILE v NON COLORATO
  - SCEGLI c_v
  - IF $c_v = c_u$ PER QUALCHE $u \in N(v)$
    - RIGETTA c_v
  - ELSE
    - c_v DEFINITIVO
    - NOTIFICA A $u \in N(v)$ LA SCELTA
    - c_u RIMOSSO DA $Palette(u)$

ANALYSIS

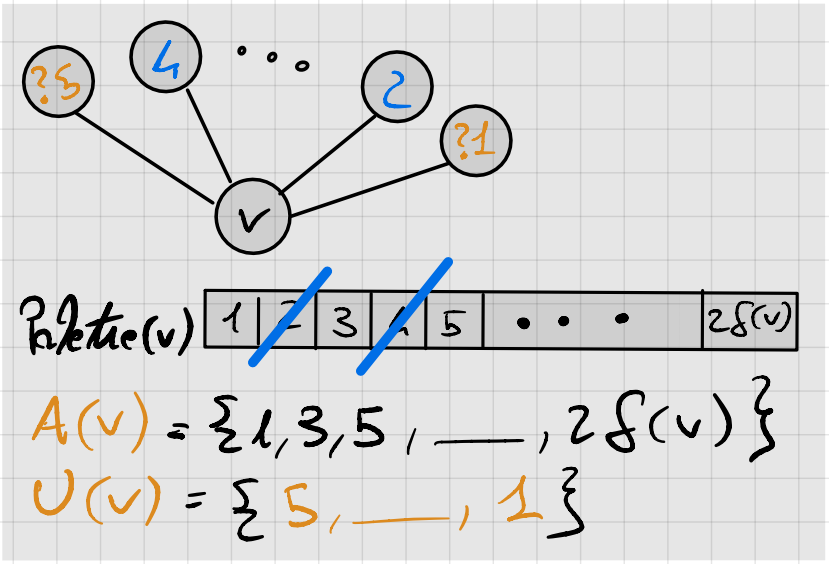
CONSIDERA v ALLA FASE k

$A(v) = \{c_v$ DISPONIBILE T/C $c_v \in Palette_k(v)\}$

$U(v) = \{c_u$ CANDIDATO T/C $c_u \in N_{NON\ COL}(v)\}$

$A'(v) = A(v) \setminus U(v)$

$\delta = |N_{NON\text{-}COL}(v)|$



SEGUE:

$|A'(v)| = |A(v) \setminus U(v)| >$

$> |A(v)| - |U(v)| \geq$

$\geq (2\delta(v) - (\delta(v) - \delta)) - \delta = \delta(v)$

$2\delta(v)$: $\max |Palette(v)|$; $\delta(v) - \delta$: $N_{COL}(v)$; δ: $|U(v)| \leq \delta$

QUINDI:

$Prob_k(v$ ACCETTA $c_v) = \frac{|A'(v)|}{|A(v)|} \geq \frac{\delta(v)}{2\delta(v)} = 1/2$

$Prob_{k=1\_2\log n}(v$ RIGETTA $c_v) \leq (1-1/2)^{2\log n} = \frac{1}{2^{2\log n}} = 1/n^2$

$Prob_{k=1_2\log n}($ALMENO UN v RIGETTA $c_v) \leq n \cdot 1/n^2 = 1/n$

$Prob_{k=1_2\log n}($OGNI v ACCETTA $c_v) \geq 1 - 1/n$

$\Rightarrow$ CON PROBABILITÀ $1 - 1/n$ L'ALGO IMPIEGA $2\log n$ FASI;

OGNI FASE IMPIEGA $O(1)$

COMPLEXITY

$O(2\log n)$ HIGH PROB

MUTEX

PROBLEMA: ACCESSO COORDINATO

ASSUMIAMO: NON-ANONYMOUS, NON-UNIFORM, ASYNCHRONOUS

UN MUTEX ALGO SPECIFICA ENTRY/EXIT SECTION, AL FINE DI GARANTIRE:

(1) MUTUAL EXCLUSION
(2) LIVENESS CONDITION

} NO-DEADLOCK, NO-LOCKOUT, BOUNDED WAITING

LA COMPLEXITY DIPENDE DAL #SHARED_VARIABLES

BAKERY ALGO

GARANTISCE MUTUAL EXCLUSION E BOUNDED WAITING

ALGO

CHOOSING$_i$ = TRUE
NUM$_i$ = max{NUM$_0$, —, NUM$_{n-1}$} + 1
CHOOSING$_i$ = FALSE
FOR $j=0,\ldots,n-1$ T/C $j \neq i$
  WAIT UNTILL CHOOSING$_j$ = FALSE
  WAIT UNTILL NUM$_j$ = 0 OR NUM$_j$ > NUM$_i$
NUM$_i$ = Ø

ENTRY
EXIT

COMPLEXITY

$2n$ R/W VARS

Lo mutual exclusion

LEMMA 1 SE P_i IN CS $\Rightarrow NUM_i > 0$

LEMMA 2 SE P_i IN CS, $NUM_k \neq \emptyset$, $k \neq i \Rightarrow NUM_k > NUM_i$

Proof

(CASO 1) P_k SCEGLIE DOPO P_i, DATO CHE NUM_i GIÀ STATO PRESO, P_k PRENDE $NUM_k > NUM_i$

(CASO 2) P_k SCEGLIE PRIMA DI P_i, P_i FORZATO A SCEGLIERE $NUM_i > NUM_k$ □

$\Rightarrow$ SUPPONENDO P_i, P_k IN CS, DA LEMMA 1 $NUM_i, NUM_k > 0$, DA LEMMA 2: $NUM_k > NUM_i \wedge NUM_i > NUM_k$ ↯

No Deadlock

P_i AL PIÙ SI BLOCCA AL SECONDO WAIT, OGNI ALTRO P_k, O ENTRA PRENDENDO $NUM_k > NUM_i$ O ESCE PRENDENDO $NUM_k = \emptyset$

Lo bounded-waiting

P_i NELL'ENTRY SECTION PUÒ ESSERE SORPASSATO AL PIÙ $n-1$ VOLTE

BOUNDED STAC 2-PROCS MUTEX ALGO

SHARED VARS $w[0], w[1]$

CODICE ASIMMETRICO P_0 SEMPRE PRIORITARIO

P_0's CODE

```
1
2
3  w[0] = 1
4
5
6  WAIT UNTILL (w[1] = 0)

1
2  w[0] = 0
```

P_1's CODE

```
1  w[1] = 0
2  WAIT UNTIL (w[0] = 0)
3  w[1] = 1
4
5  IF w[0] = 1 GOTO 1
6

1
2  w[1] = 1
```

↳ mutual exclusion

P_i ENTRA $\Leftrightarrow$ $w[i] = 1$ E $w[i-1] = 0$

↳ SE P_1 IN CS E P_0 ALLA RIGA 5 $\Rightarrow w[0] = w[1] = 1$

↳ SE P_0 IN CS E P_1 ALLA RIGA 2 $\Rightarrow w[0] = 1$, $w[1] = 0$

↳ no deadlock

SE P_0 SETTA $w[0] = 1$, P_1 COSTRETTA ALLA RIGA 5 A SETTARE $w[1] = 0$

↳ no lockout (sì)

SE P_0 SETTA $w[0] = 1$ MENTRE P_1 TRA LE RIGHE 3 e 5 IN CONTINUAZIONE $\Rightarrow$ P_1 BLOCCATO

↳ (NO)

INTRODUCIAMO PRIORITY E UNIFICHIAMO I CODICI

P_i's CODE

```
1  w[i] = 0
2  WAIT UNTIL( w[1-i] = 0 OR PRIORITY = i )
3  w[i] = 1
4  IF(PRIORITY = i-1)
5      IF( w[i-1] = 1 ) GOTO 1
6  ELSE
7      WAIT UNTIL( w[i-1] = 0 )
8  PRIORITY = i-1
9  w[i] = 0
```

ANALYSIS

↳ mutual exclusion

SUPPONIAMO P_0, P_1 IN CS

$w[0] = w[1] = TRUE$, SETTARE ALLA RIGA 3

UNO DEI DUE DEVE BLOCCARSI A RIGA 5 o 6

↳ no deadlock

SUPPONIAMO DEADLOCK E PRIORITY = 0

P_0, P_1 BLOCCATI A RIGA 2, $w[0] = w[1] = 0$

P_0 HA PRIORITY => SUPERA RIGA 2 => VA A RIGA 3

$w[0] = 1$

P_0 SUPERA RIGA 6 PERCHÉ $w[1] = 1$

P_0 IN CS

P_0 ESCE DA CS => PRIORITY = 1 E $w[0] = 0$

P_1 SORPASSA RIGA 2 ⊥

↳ no lockout

ASSUMI P_0 BLOCCATO A RIGA 2

P_1 PROSEGUE, USCENDO DA CS SETTA PRIORITY = 0

P_0 PROSEGUE ⊥

↳ bounded waiting (NO)

P_0 SORPASSABILE ∞ TRA RIGA 2 E RIGA 3

BOUNDED SPACE M-PROCS MUTEX ALGO

ASSUMIAMO $m = 2^k$, $k > 1$

COSTRUIAMO UN TOURNAMENT TREE DI $m-1$ NODI

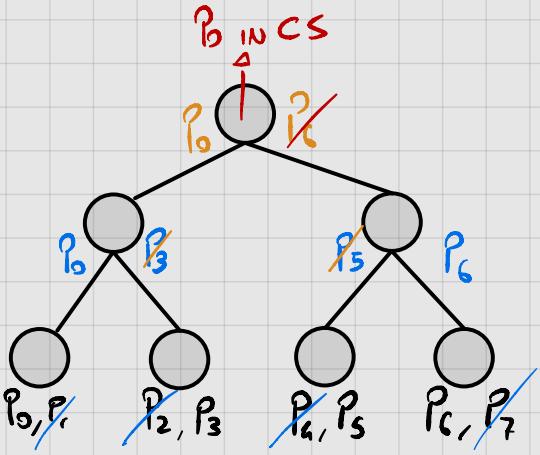
ASSOCIAMO A OGNI NODO UN IL 2-PROCESSOR MUTEX ALGO

$m = 2^3 = 8$

ASSOCIAMO I PROCESSORI ALLE FOGLIE DA SX A DX

SE $m < 2^k$ AGGIUNGIAMO DUMMY LEAVES

RICORSIONE DEL 2PROCS ALGO A PARTIRE DALLE FOGLIE



ANALYSIS COME 2PROCS ALGO

COMPLEXITY

$3(m-1)$ BOOLEAN R/W

- m → # VARS DEL 2-PROCS ALGO
- $(m-1)$ → # NODI

CONSENSUS

FAILURES { LINK, CRASH, BYZANTINE

PROPRIETÁ ALGOS (PER P_i NON-FAULTY) {
- TERMINATION P_i SCEGLIE $y \in Y$
- AGREEMENT TUTTI P_i SCELGONO $y \in Y$
- VALIDITY SE TUTTI P_i HANNO STESSO INPUT $x \in X$, ALLORA SCELGONO STESSO $y \in Y$

LINK FAILURE

ESISTONO ISTANZE DI INPUT PER LE QUALI IL CONSENSO NON È RAGGIUNGIBILE IN CASO DI LINK-FAILURE

NEGATIVE RESULTS (2 GENERALI)

SIA $\mathcal{A}$ LO SHORTEST PROTOCOL PER IL CONSENSO.

SE UN MSG m VIENE PERSO, E IL CONSENSO RAGGIUNTO COMUNQUE

=> $\mathcal{A}$ NON È LO SHORTEST PATH ⊥

PROCESSOR FAILURE

NEGATIVE RESULTS

NEL CASO ASINCRONO IMPOSSIBILE CONSENSO ANCHÉ CON UN SOLO PROC-FAIL

POSITIVE RESULTS

SINCRONO E CLIQUE TOPOLOGY

COMPLETO (=> NON UNIFORME)

SYNC START

LOW BOUND

TH OGNI F-RESILIENT TO CRASH-FAIL CONSENSUS ALGO RICHIEDE ALMENO $f+1$ ROUND

Proof

SE BASTASSERO MENO DI $f+1$ ROUND, TUTTI DOVREBBERO AVERE STESSA KNOWLEDGE

WORST CASE

UN P_i A ROUND FALLISCE INVIANDO VAL_i SOLO A UN ALTRO P_j

P_j È QUELLO CHE FALLISCE AL ROUND DOPO

AL ROUND $f+1$ P_i È L'UNICO AD AVER RICEVUTO IL VAL x DAL PROC CRASHATO

SE P_i SCEGLIE x È L'UNICO A FARLO, POICHÉ SOLO LUI LO CONOSCE

⇒ f ROUND NON SONO ABBASTANZA (SE P_i AVESSE BROADCASTATO PRIMA DI SCEGLIERE X, LO AVREBBERO SCELTO TUTTI) □

SIMPLE ALGO (FAULT FREE)

FOR OGNI P_i

- BROADCAST VAL_i
- LEGGE INPUTS
- SCEGLIE min VAL

// NON FUNZIONA CON CRASH-FAIL

// 1 ROUND (G COMPLETO)

F-RESILIENT TO CRASH ALGO

FOR OGNI P_i

ROUND 1 BROADCAST VAL$_i$

ROUND 2 TO F+1 BROADCAST OGNI VAL RICEVUTO

END ROUND F+1 SCEGLIE MIN VAL

LEMMA 1 END F+1 $\Rightarrow$ OGNI P_i CONOSCE STESSI VALS

Proof

ASSUMI LEMMA FALSO

SIA x VALORE CONOSCIUTO DA $P' \subseteq P$ A END F+1

P' RICEVE x A F+1 MA ESSENDO QUESTO IL ROUND SENZA CRASH FAIL, x BROADCASTATO

$P' = P$ $\perp$ □

CORRETTEZZA

↳ AGREEMENT SYNCH START -> LA SAME-KNOWLEDGE DEL ROUND F+1 NON CAMBIA

↳ VALIDITY IL VAL SCELTO FA PARTE DEI VAL IN INPUT

PERFORMANCE

$O(n \cdot n \cdot k) = O(n^3)$

n #PROCS

n #INPUT, AL PIÙ n MANDATI DA P_i

$O(\int_T 1) = \Delta \leq n$

BYZANTINE FAILURE

LOWER BOUND

TH OGNI F-RESILIENT TO BYZANTINE FAIL CONSENSUS ALGO RICHIEDE ALMENO $f+1$ ROUND

Proof SEGUE DAL LOWBOUND DEL CRASH-FAIL □

F-RESILIENT TO BYZ-FAIL ALGO

ID's DISTINTI (NON-ANONYMOUS), NON UNIFORM

$2(f+1)$ ROUND PER n PROCS DI CUI $n/4$ BYZANTINE

↳ $f \leq n/4 \Rightarrow n \geq 4f+1$

$f+1$ FASI DA 2 ROUND

UN DIVERSO KING A FASE $\Rightarrow$ ALMENO UNO NON FAULTY

ALGO

FOR $k=1 \_ f+1$

- BROADCAST v_i
- SIA a IL PIÙ FREQUENTE RICEVUTO (MAJORITY VAL)
  - SIA $1 \leq m_i \leq n$ #OCCORRENZE (MAJORITY)
    - $v_i = a$

ROUND 1

- KING P_k BROADCASTA v_k
- FOR OGNI P_i
  - IF v_i T/C $m_i < n/2 + 1 + f$ (WEAK MAJORITY)
    - $v_i = v_k$
  - ELSE
    - v_i RESTA a

ROUND 2

- P_i SCEGLI IL SUO v_i

END $f+1$

CORRETTEZZA

LEMMA 1 END k DOVE KING NON-FAULTY $\Rightarrow$ OGNI P_i NON FAULTY SCEGLIE v_k

Proof

- CASO 1

  SE OGNI P_i HA WEAK MAJ A END 1 $\Rightarrow$ TUTTI SCELGONO v_k AL ROUND 2

- CASO 2

  SE UN P_i HA STRONG MAJ $v_i = a$

  ALMENO $n/2+1$ HANNO BROADCASTATO a $\rightarrow$ TUTTI P_i NON FAULTY HANNO RICEVUTO a (KING COMPRESO)

  TUTTI SCELGONO a CON STRONG MAJ (CHE SIA IL PROPRIO O QUELLO DEL KING

□

LEMMA 2 SE $a = v_i \; \forall P_i$ ALLA FASE k $\Rightarrow$ I v_i NON CAMBIANO

Proof

AL PIU f BYZ. PROCS $\rightarrow n-f$ non faulty

$f \leq n/4 \Rightarrow n-f > n/2+f$

DOPO k a AVRÀ SEMPRE STRONG MAJ ($> n/2+f$)

- DI SCEGLIERANNO SEMPRE LO STESSO VALORE

□

- **agreement** ESSENDOCI SEMPRE UNA FASE SENZA BYZ. FAIL
  - DAL LEMMA 1 IN QUESTA, TUTTI SCELGONO STESSO a
  - DAL LEMMA 2 a NON CAMBIA PIÙ

- **validity** SE OGNI P_i HA IN INPUT STESSO a, QUESTO HA STRONG MAJORITY ($n-f$), ED È SCELTO. DAL LEMMA 2 TALE COMMON v_i NON CAMBIA.

PERFORMANCE

$n > 4f$ #PROCS, $2(f+1)$ #ROUNDS

$\Rightarrow O(n^2 \cdot f) = O(n^3)$ #MSGS

- n^2: n msgs in ROUND 1 DA P_i NON FAULTY
- f: ALPIÙ $n/4$
- $n-1$ msgs DA NON-FAULTY KING AL ROUND 2

RANDOMIZED BYZANTINE CONSENSUS ALGO

C'È UN PROC. q SEMPRE AFFIDABILE

A OGNI ROUND q LANCIA UNA MONETA

C'È v_i PREFERRED VALUE DI P_i

$\forall$ CM/$\forall$

THRESHOLD $\begin{cases} L = 5 \text{ M/}\forall \\ M = 6 \text{ M/}\forall \\ H = 7 \text{ M/}\forall \end{cases}$

ALGO

P_i AD OGNI ROUND

- BROADCAST v_i
- RICEVE $v_j \; \forall \; P_j$
- SIA MAJ_i IL MAJORITY VAL E $TALLY_i$ LA SUA FREQUENZA
- RICEVE COIN-OUTCOME DA q
- IF TESTA
  - THRESHOLD = L
- ELSE
  - THRESHOLD = M
- IF TALLY $\geq$ THRESHOLD
  - $v_i = MAJ_i$
- ELSE
  - $v_i := \emptyset$
- IF TALLY $> H$
  - STOP

// CASI A FINE ROUND

// (1) TERMINATION

// (2) P_i, P_k T/C $maj_i \neq maj_k$

// (3) maj_i STESSO $\forall P_i$

// $\Rightarrow$ TUTTI PORTANO AL CONSENSO

ANALYSIS

(1) TERMINATION

TALLY$_i$ > G PER maj$_i$:

DATO CHE $f < n/3$ I VOTI PER maj$_i$ DA NON-FAULTY SONO TALLY$_i$ $- f > M$

OGNI NON FAULTY P_k T/C V_k = maj$_k$ = maj$_i$ E TALLY$_k$ > M $\Rightarrow$ CONSENSO RAGGIUNTO

LEMMA SE ALL'INIZIO DI UN ROUND V_k STESSO $\forall P_k$: L'ALGO TERMINA E RAGGIUNGE CONSENSO

Proof

ASSUMIAMO $|TALLY_i - TALLY_k| > f$

SIANO f', f'' #CORRUPTED VALUES

SIANO $TALLY_i^+$, $TALL_k^+$ #TRUSTED VALUES

$\hookrightarrow$ TALLY$_i$ $- f' =$ TALLY$_i^+$

$\Rightarrow |TALLY_i - TALLY_k| = |TALLY_i^+ + f' - TALLY_k^+ - f''|$

$= |f' - f''| > f$ $\perp$

$\hookrightarrow$ LA DIFFERENZA DI $f' \le f$, $f'' \le f$ NON PUÒ DARE UN VALORE MAGGIORE DI f

□

(3) maj$_i$ STESSO $\forall P_i$

SOTTOCASI

- TALLY$_{min}$
  - $< L$ E THRESHOLD $= M$
  - $\ge L$ E " $= L$ } 1/2P, GOOD
  - $< L$ E " $= L$
  - $\ge L$ E " $= M$ } 1/2P, BAD

- CASO $< L$ → $TALLY_k \leq TALLY_i + f < L + f \leq \frac{6m}{8} = H$
  ⇒ $V_i = V_k = 0$
- CASO $\geq L$ → $TALLY_k \geq +TALLY_{min} \geq L$
  ⇒ $V_k = V_{min} = maj_{min}$

↳ IN ENTRAMBI I CASI L'ALGO TERMINA AL PROSSIMO ROUND PERCHÉ TUTTI i V_i SONO UGUALI

↳ NEGLI ALTRI DUE CASI L'ALGO NON TERMINA AL PROSSIMO ROUND

PERFORMANCE

$O(\log m)$ #ROUNDS

↳ P(TERMINI AL PROSS ROUND) = $1/2$

↳ P(NON TERMINI IN LOG m ROUNDS) = $(1/2)^{\log m} = 1/m$

↳ P(TERMINARE IN LOG m ROUNDS) = $1 - 1/m$

$O(m^2)$ #MSG A ROUND

$O(m^2 \log m)$ #MSGS

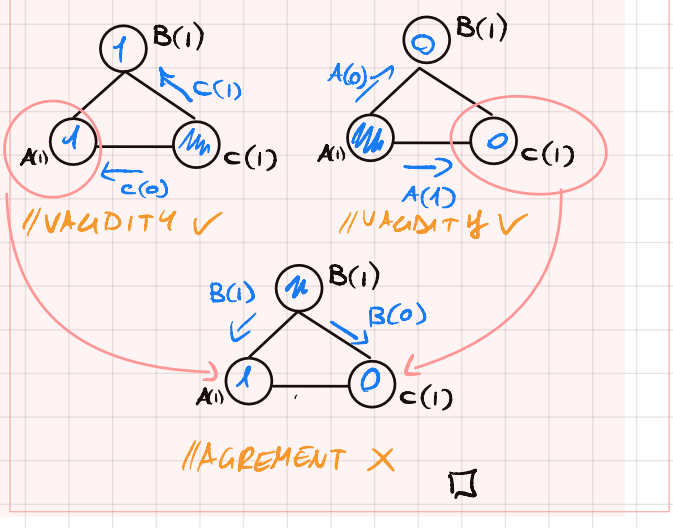
IMPOSSIBILITY RESULT

TH NON ESISTE f-RESILIENT TO BYZ-FAIL CONSENSUS ALGO PER m PROCS SE $f = m/3$

Proof

- LEMMA 1 L'ALGO NON ESISTE PER $m = 3$, $f = 1$

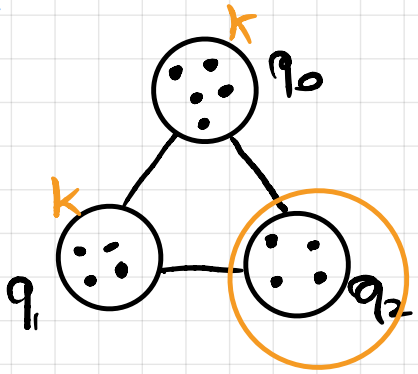
Proof



□

- ASSUMIAMO ESISTA ALGO A PER $f = n/3$, $n = 3$
- USIAMOLO PER RISOLVERE UN ISTANZA CON $n=3$, $f=1$
  - SIA $n = 3f$
  - SIA $P = \langle P_0, \_, P_n \rangle$
  - SIA $Q = \langle q_0, q_1, q_2 \rangle$ T/C $q = \langle P_0, \_, P_{n/3} \rangle$
  - USIAMO A SUL SOTTOINSIEME q_i
  - ASSUMIAMO q_2 FAIL PERCHÉ $P_i \in q_2$
  - SUPPONIAMO q_1, q_2 SCELGONO k

$\Rightarrow$ CONSENSO RAGGIUNTO CON $n = 3$ E $f = 1$ ⊥ (LEMMA 1)



□

EXPONENTIAL TREE ALG

$n = 3f + 1$

$f+1$ ROUNDS

OGNI P_i HA UN TREE ASSOCIATO

OGNI NODO DELL'ALBERO RAPPRESENTA UN P_S, E HA UNA SEQUENZA ASSOCIATA

ROOT ($h = f+1$) → λ

n - #CHILDREN(ROOT) → $0, \_, n-1$

$n-1$ - #CHILDREN(CHILD(ROOT)) → $i : 0, \_, i : n-1$ TRANNE $i:i$

$n-d$ - #CHILDREN(NODO_A_LV-d) → $i_1 : \_ : i_d : i_{0,\_} \; i_1 : \_ : i_d : i_{n-1}$

$i_1 : i_2 : \_ : i_d$

LV $f+1$ ($h = 0$) → $i_1 : i_2 : \_ : i_{f+1}$

IL LV i È RIEMPITO AL ROUND i

AL ROUND $f+1$ È COMPUTATA LA DECISIONE (BYZ-UP)

ROUND 1

INPUT -> ROOT DI P_j

BROADCASTATO A TUTTI i PROCS

P_i STOCA x NEL SUO $LV1$ AL NODO j

ROUND 2

P_j BROADCASTA A TUTTI IL SUO $LV1 = \{x_1, \_, x_{n-1}\}$

P_i SCARTA x_j E STOCA x_k NEL SUO $LV2$ AL NODO $k:j$

↳ NON ∃ A LV2 IL NODO $j:j$

ROUND $d \geq 2$

P_j BROADCASTA LV_{d-1} CIOÈ MANDA $n(n-1)\dots(n(d-2))$ MSGS

P_i STOCA x NEL LV_d AL NODO $i_1 : i_2 : \_ : i_{d-1} : j$

ROUND $f+1$

COMPUTA RICORSIVAMENTE $resolve(\pi)$ $\forall$ NODO π

$$resolve(\pi) \begin{cases} \pi \text{ SE } \pi \text{ FOGLIA} \\ MAJORITY \{ \pi' \text{ T/C } \pi' = CHILD(\pi) \} \end{cases}$$

CONSISTENZA DI resolve( )

LEMMA 1 SE $n \geq 3f$ E P_j, P_i NON FAULTY

$\Rightarrow$ resolve($\tilde{n}_i[j]$) = resolve($\tilde{n}_j$)

Proof

INDUZIONE SU h

BASE

$\tilde{n}_i$ FOGLIA, $h = 0$

$\Rightarrow P_i$ STORE IN $\tilde{n}_i[j] = \tilde{n}_j$ PROPRIO $\tilde{n}_j$ CIOÈ IL VALORE MANDATO DA P_j AL ROUND $f+1$

IND

$\tilde{n}_i$ NON FOGLIA, $h > 0$

$\Rightarrow \tilde{n}_i$ HA $n-$ FIGLI

↳ $n > 3f \Rightarrow n - f > 2f \Rightarrow \tilde{n}_i$ HA LA MAJORITY DI NON FAULTY CHILDREN (IL CUI ULTIMO DIGIT È L'ID DI UN PROC. NON FAULTY)

↳ SIA $\tilde{n}_k$ $k = \tilde{n}_j[j,k]$ FIGLIO DI $\tilde{n}_i$ DI $h-1$ TC P_k NON FAULTY $\Rightarrow P_j$ (NON FAULTY MANDA IL VALORE CORRETTO v E P_k LO STORA CORRETTAMENTE IN $\tilde{n}_k = \tilde{n}_j[j]$)

$\Rightarrow$ PER INDUZIONE

resolve($\tilde{n}_i = \tilde{n}_j[j]$) $= v =$ resolve($\tilde{n}_k = \tilde{n}_j[j,k]$)

$\Rightarrow$ OGNI NON FAULTY CHILD DI $\tilde{n}$ RISOLVE $v \Rightarrow \tilde{n}$ ANCHE

□

VALIDITY

SUPPONIAMO ALL INPUT V

SE P_j NON FAULTY $\Rightarrow$ P_i STORA V IN J

(LEMMA 1)

ESSENDO MAJORITY $\geq 2f$ DI NON FAULTY $\Rightarrow$ P_i ROOT RESOLVE V

AGREEMENT

Def COMMON $\tilde{\pi}$ SE OGNI NON FAULTY STESSO resolve($\tilde{\pi}$)

LA ROOT È COMMON?

↳ LEMMA 1 CATTURA TUTTI i NODI CON ULTIMO DICIT UGUALE A PROCESSORE NON FAULTY

Def $\tilde{\pi}$ HA COMMON FRONTIER SE OGNI PATH FINO ALLE FOGLIE HA UN COMMON NODE

LEMMA 2 SE $\tilde{\pi}$ HA COMMON FRONTIER $\Rightarrow$ $\tilde{\pi}$ È COMMON

Proof

INDUZIONE SU h, $\tilde{\pi}$ HA COMMON FRONTIER

BASE

$\tilde{\pi}$ FOGLIA, $h=0$

$\Rightarrow$ ESSENDO SOLO $\tilde{\pi}$ NEL PATH, $\tilde{\pi}$ È COMMON

INDUZIONE

$\tilde{\pi}$ NON FOGLIA, $h>0$

↳ ASSUMIAMO ABBIA NON SIA COMMON

$\Rightarrow$ OGNI FIGLIO $\tilde{\pi}'$ HA COMMON FRONTIER

$\tilde{\pi}'$ HA $h-1$

$\Rightarrow$ PER INDUZIONE $\tilde{\pi}'$ COMMON POICHÉ LE FOGLIE LO SONO, E i PADRI PRENDONO IL VALORE DEI FIGLI

↓ □

AGREEMENT

$f+1$ # NODI TRA ROOT - LEAF

I NODI SUL PATH HANNO LAST-DIGIT DISTINTO

ALMENO UN NODO NON FAULTY SUL PATH

$\Rightarrow$ É COMMON

↳ LEMMA 1

$\Rightarrow$ LA ROOT HA COMMON FRONTIER $\Rightarrow$ É COMMON

↳ LEMMA 2

TERMINATION $f+1$ ROUNDS

COMPLEXITY

ESPONENZIALE IN n

- ROUND 1

$O(n^2)$

$\#O(n)\ O(n)$ (NON FAULTY, #MSGS)

- ROUND $2 \le d \le f+1$

$O(n^{d+1})$

$\#O(n\ n\ n(n-1)\_(n-(d-2))$

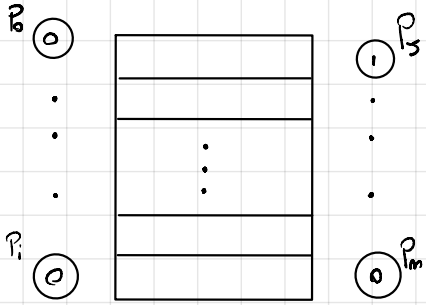
(n: NON FAULTY; $n-1$: PROCS A CUI MANDARE IL LV $d-1$; $n(n-1)\_(n-(d-2))$: #NODI AL LV $d-1$)

$\Rightarrow O(n^2)+O(n^3)+\_+O(n^{f+2}) = O(n^{n+2})$

($f = O(n)$)

CONSENSUS IN SHARED MEMORY

$P_0, \dots, P_{m-1}$ POSSONO CRASHARE



Def WAIT FREEDOM IN ASYNC SYS
P_i DEVE POTER COMPLETARE L'ESECUZIONE ANCHE SE GLI ALTRI CRASHANO

↳ IN MPS IL CONSENSO NON È RAGG. ANCHE PER UN SOLO CRASH-FAIL

Def CONSENSUS NUMBER MASSIMO NUMERO DI PROCESSI PER CUI UN SINGOLO TIPO DI SHARED VAR PUÒ ESSERE USATA PER RAGGIUNGERE WAIT FREE CONSENSUS

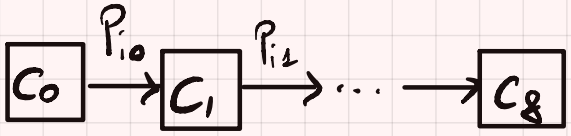
TH IL CONSENSUS NUM PER R/W È 1

Proof

PER $n=1$ BANALE

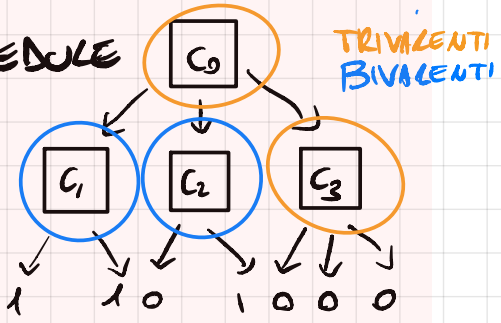
PER $n \geq 2$ MOSTRIAMO CHE OGNI ALGO NON TERMIN

SYSTEM CONFIGURATION
SET DI TUTTE VARS (SHARED + LOCAL)



TREE OF CONFIGURATION

↳ IL PATH SCELTO DIPENDE DALLO SCHEDULE DI CRASH E AZIONI



VALENZA
INSIEME DI VALORI DECIDIBILI DA OGNI NON FAULTY IN C_1 RAGGIUNGIBILE DA C CON UN ESECUZIONE AMMISSIBILE

NOTA VALIDITY SE GLI INPUT (CIOÈ C_0) SONO UGUALI ⇒ UNIVALENTE

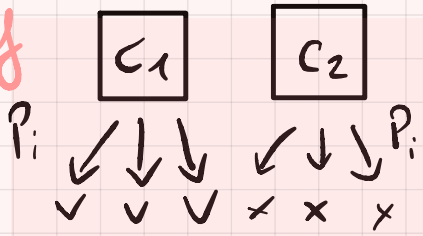
PER FAR SI CHE L'ALGO TERMINI (CON CONSENSO) PRIMA DI C_f DEVE ESSERE C_i UNIVALENTE

SIMILAR CONFIGURATION

STESSE SHARED, DIVERSE LOCAL $C_1 \overset{P_i}{\approx} C_2$

LEMMA SE ESISTONO C_1, C_2 UNIVALENTI T/C $C_1 \overset{P_i}{\approx} C_2 \Rightarrow$
SE C_1 V-VALENTE $\Rightarrow$ C_2 V-VALENTE

Proof



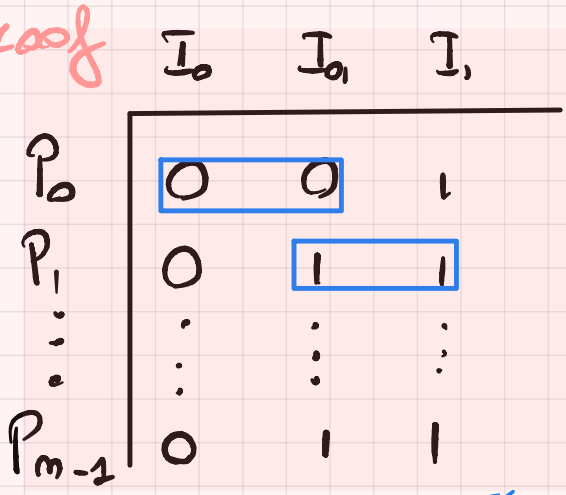
$\Rightarrow C_1 \overset{P_i}{\approx} C_2 \Rightarrow V = X$

↑ STESSE SHARED

□

LEMMA $\exists C_0$ BIVALENTE

Proof

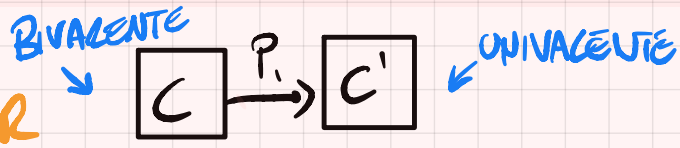


$\Rightarrow I_{01}$ NO 0-VALENTE $\Rightarrow I_{01} \overset{P_0}{\approx} I_0$

$\Rightarrow I_{01}$ NO 1-VALENTE $\Rightarrow I_{01} \overset{P_1}{\approx} I_1$

□

CRITICAL PROCESSOR



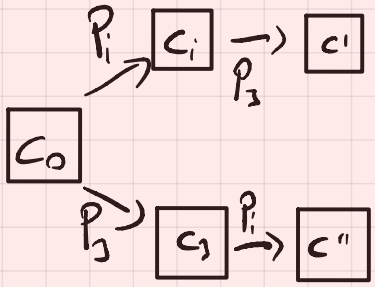
LEMMA SE C BIVALENTE C'È ALMENO UN P_i NON CRITICAL

Proof

ASSUMIAMO TUTTI CRITICAL

$\Rightarrow$ NON POSSONO AVERE STESSA VALENZA V SENNO C_0 CRITICAL

$\Rightarrow$ ESISTONO P_i, P_j CON DIVERSA V



CASI

- R/R DIVERSE SHARED $\} \Rightarrow C' = C''$
- R/R STESSE " "
- R/W " " $\} \Rightarrow C' \overset{P_i}{\approx} C''$
- W/W " "

$\Rightarrow$ ALMENO UNO NON-CRITICAL □

$\Rightarrow$ POSSIAMO COSTRUIRE UN PATH DI SOLI NON CRITICAL $\Rightarrow$ L'ALGO PUÓ NON TERMINARE □

HASH FUNCTION

Hash Functions (Attacks)

Preimage attack:
given h, find m such that H(m) = h.

Second preimage attack:
given m_1, find $m_2 \neq m_1$ such that $H(m_1) = H(m_2)$.

Birthday attack:
find m_1 and $m_2 \neq m_1$ such that $H(m_1) = H(m_2)$.

CRYPTOGRAPHIC HASH

- Is a one-way function: avoids pre-image attacks.
- Resistant to second pre-image attacks.
- Collision resistant; avoids birthday attacks.
- A small change to the input produces a big change in the output.
- Resistant to other attacks (e.g., length extension).

DISTRIBUTED LEDGER PROBLEM : LEDGER CONTIENE TRANSAZIONI, TUTTI POSSONO AGGIUNGERE TRANSAZIONI (NO CENTRAL AUTHORITY)

LEDGER PROBLEM

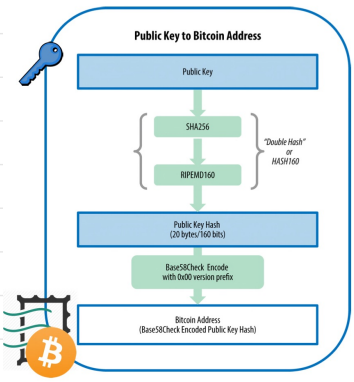
RISOLUZIONE CON DIGITAL SIGNATURE

OGNI NODO HA $\{K_R, K_{PUB}\}$
LA CENTRAL AUTHORITY HA LA CHIAVE PUB. DI TUTTI E TUTTI HANNO LA SUA

BITCOIN ARCHITECTURE

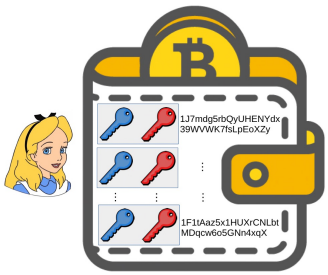
BITCOIN ABBIAMO CON PAGAMENTI, CAMBIO DI VALUTA, MINING

Bitcoin Address



Bitcoins are "owned" by an address (a public key).
They can be spent by whoever controls the corresponding private key.

Bitcoin Wallet

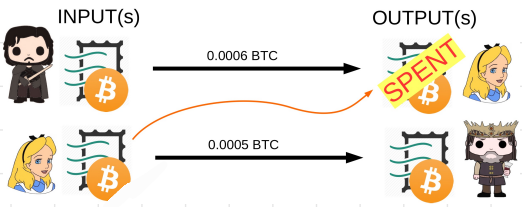


} SEEDED / RANDOM
↓ determ. ↓ non determ

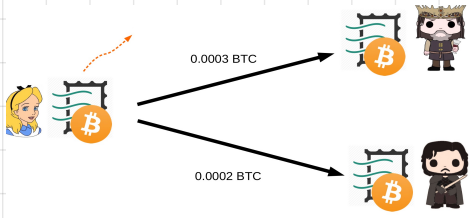
TRANSACTION

NO DOUBLE SPENDING

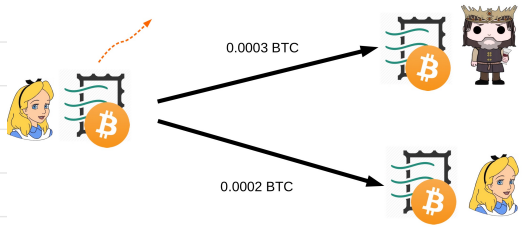
(PROVA CHE HAI I SOLDI)



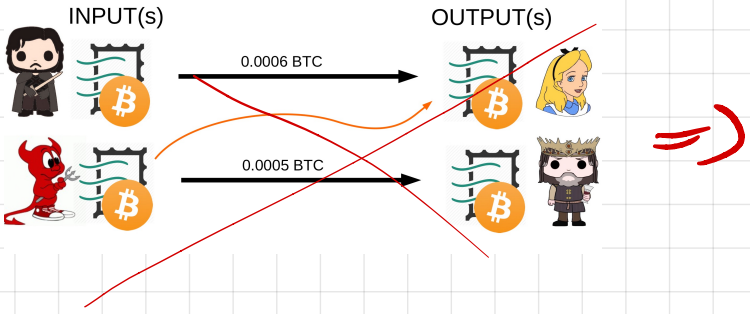
MULTIPLE OUTPUTS



NO CHANGE



DIGITAL SIGNATURE



A Bitcoin transaction:

1. One or more senders.
2. One or more receivers.
3. The amount of BTC (Bitcoins) transferred from each sender to each receiver.
4. A proof of ownership of the coins being transferred, in the form of a pointer back to most recent transactions involving the transferred coins.
5. A transaction fee, paid by the sender to the authorizer of the transaction.

A transaction is valid if:

1. It has been cryptographically signed by all of the senders.
2. The senders really do own the coins being transferred.

This can be verified using the senders' public keys.

This can be verified as follows:

- transactions are broadcast to all other users (through a peer-to-peer network);
- all users keep track of all transactions that have ever been authorized;
- thus, everyone knows everyone's current balance

BLOCKS

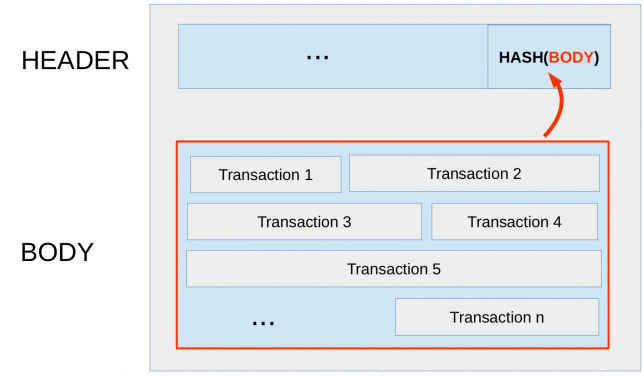
TRANSAZIONI RAGGRUPPATE IN BLOCCHI DELLA SIZE DI 1MB

Transactions are added to the ledger in groups, known as blocks.

A block contains:

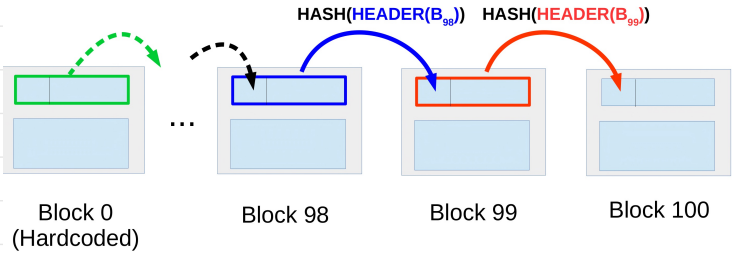
1. One or more transactions.
2. A hash of the previous block.
3. A nonce. (I.e., a bunch of bits that can be set arbitrarily.)

This imposes a natural linked list-type structure on the ledger:
- the predecessor of a block b_2 being the block b_1 whose hash matches the hash stored in b_2.



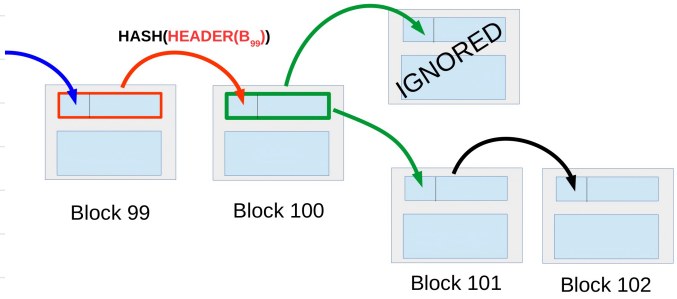
BLOCKCHAIN

- OGNI BLOCCO SALVA HASH(PADRE) NELL'HEADER



FORK

SHORT BRANCHES IGNORATI POI



LO TROVA PIÙ RICOSTRUIRE LA STORIA DELLE TRANSAZIONI SOLO CON IL 50% DEL POTERE COMPUTAZIONALE DEL NETWORK

Malicious peers who want to modify past blocks have to work harder than honest peers who want append blocks.

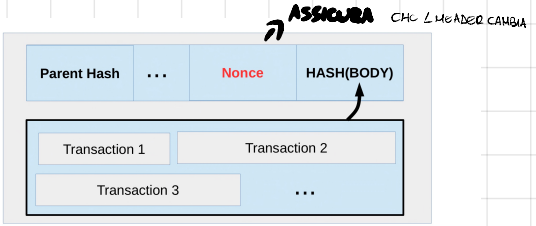
A block B is only accepted by the network iff
HASH(HEADER(B)) ≤ **TARGET**

TARGET dynamically updates so that the average time to find a valid block is around 10 minutes

MINING

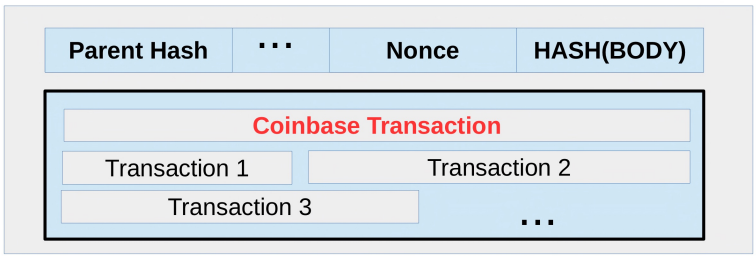
MEMPOOL TRANSAZIONI NON APPARTENENTI AD ALCUN BLOCCO
MINER SCEGLIONO UN SUBSET DI QUESTE E LE METTONO NEL BODY DEL NUOVO BLOCCO

- GENERANO UN HEADER FINCHÉ HASH(Header) ≤ **TARGET**



GUADAGNO SOLDI!!

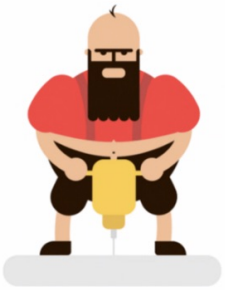
- AGGIUNGONO UNA COIN BASE TRANSACTION AD UN PROPRIO INDIRIZZO
  LA TRANSAZIONE È UGUALE ALLA SOMMA DI:
  - → BLOCK SUBSIDY: SOMMA DEL PATH FINO AL BLOCCO B
  - → TRANSACTION FEES SOMMA DEL SOTTOINSIEME DI TRANSAZIONI AGGIUNTO



Block Rewards and Bitcoin Mining

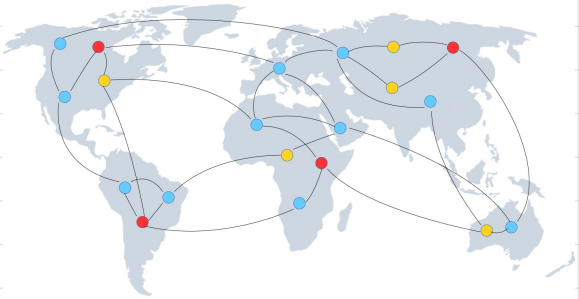
Bitcoin mining: the process of finding new valid blocks.

A *miner*:
- chooses a subset of the transactions;
- inserts the hash of the current last block;
- arbitrarily set the bits in the nonce (and hope that the resulting block is valid).



COME SELEZIONO LE TRANSAZIONI DA AGGIUNGERE? -> KNAPSACK (NP-HARD)

The Bitcoin Network



Full Nodes: have a local copy of the entire blockchain. Can directly verify transactions. Can send/receive bitcoins.

Lightweight (SPV) nodes: No local copy of the blockchain. Can indirectly verify transactions. Can send/receive bitcons.

Miners: work to extend the blockchain, mint bitcoins, and get rewarded.

→

- SPV Nodes only download block headers (80B/block).
- Complete knowledge of **which** blocks are in the blockchain
- No knowledge on **what** the block contents (transactions) are.

Some issues:
- How do new blocks get added to the blockchain?
- Who can do it?
- Why should they bother?
- How can we make sure that everybody agrees on the contents of the blockchain?

Two key ingredients:
1. Any user can authorize a block. Bitcoin incentivizes users to do authorizations through explicit monetary rewards (in BTC, naturally).
2. Authorizing a new block of transactions involves a proof of work, meaning that the authorizer has to solve a computationally difficult puzzle.

→

A block b is valid if $h(b)$ is sufficiently close to 0.

h: pre-agreed upon hash function (currently, SHA-256)

the leading l bits of $h(b)$ should all be 0, where l is a parameter

a block contains:
transactions, the hash of the previous block, the nonce

has to be set properly set in order to make the block valid

parameter l chosen to keep the rate of valid block creation roughly every ten minutes

Bitcoin Mining Protocol:

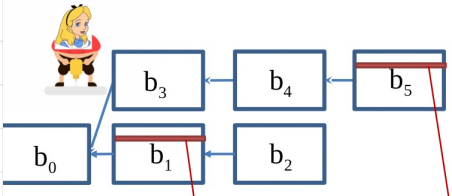
- work on the next block to be added to the longest chain
- announce the solved block as soon as you get it

Does a miner have convenience to follow the protocol?

The Double-Spend Attack

Idea: miners deliberately create forks.

Assumption: Bob only ships the purchased goods to Alice once another block b_2 has been appended to b_1.



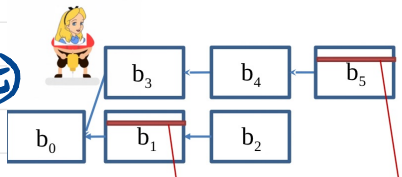
α: fraction of the computational power controlled by Alice

probability of success: α^3

The Double-Spend Attack

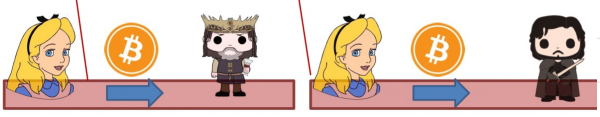
Idea: miners deliberately create forks.

Assumption: Bob only ships the purchased goods to Alice once k other blocks have been appended to b_1.



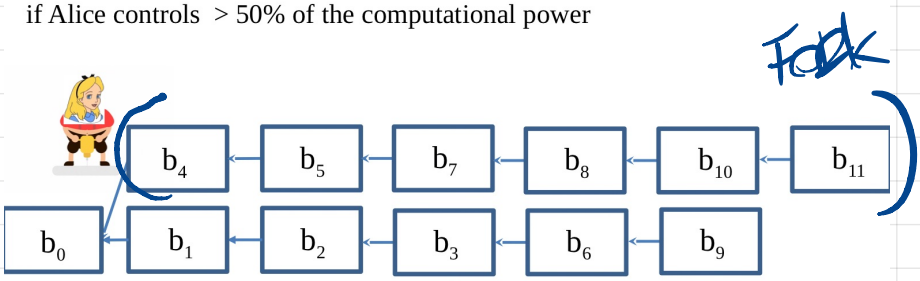
α: fraction of the computational power controlled by Alice

probability of success: α^{k+2}



The 51% Attack

if Alice controls > 50% of the computational power



remark:
Bitcoin is not intended to function when a single entity controls more than 50% of the computational power

Alice will eventually build the longest chain (with probability 1)

ASPETTI DELLA BLOCKCHAIN

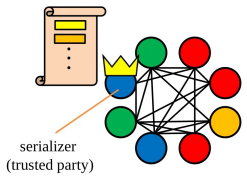
• DISTRIBUTED LEDGER

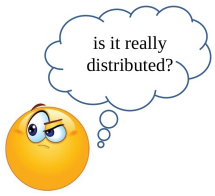
= node of the network

= blockchain

= block (of transactions)

SEMPLICE SOLUZIONE

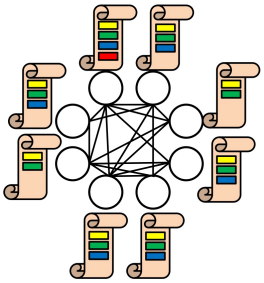




- what if the serializer fails?
- what if the serializer is not honest?

} *fault tolerance*

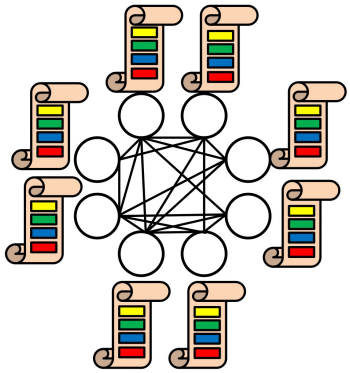
BETTER SOLUTION



Consistency:
all nodes always agree on the current state of the ledger

Eventual consistency:
all nodes eventually agree on the current state of the ledger (if no new updates are issued)

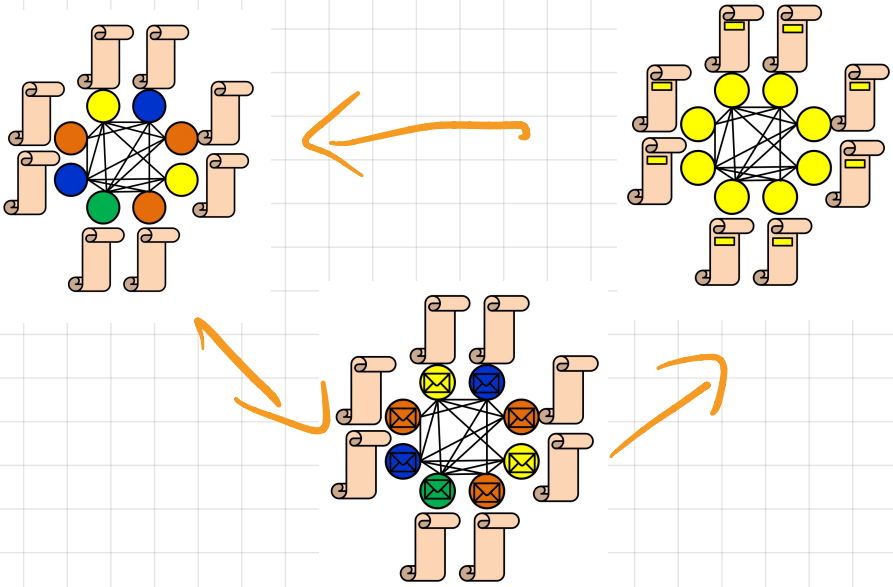
DOPO ABBASTANZA TEMPO →



• LEDGER CON CONSENSO RIPETUTO

repeat:

- each node supports its command
- exchange messages to get an agreement on the winning command
- every node updates its (local) ledger with the winning command

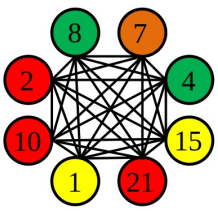


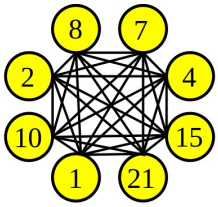
Consensus Problem

a set of n nodes
each node has:
unique ID
a color in {●●●…●}
an underlying communication graph G

Goal: a *distributed protocol* guaranteeing

Termination (protocol eventually ends)
Agreement (monochromatic final configuration)
Validity
the *winning color* is initially supported by some node





→ MEASURE

- # of messages
- size of the messages
- # of rounds (sync. model)

SYNCHRONOUS

- shared clock
- computation proceeds in *rounds*
- messages sent in a round arrive in the next round
- in a round, a node receives mgs & computes & sends msgs

ASYNCHRONOUS

- no shared clock
- no rounds
- messages arrive in the finite but unbounded time

SIMPLE SOLUTION : LEADER ELECTION (CON LEADER CHI RISOLVE IL BLOCCO)

FAULT TOLERANCE

CRASH FAILURE

Theorem (1985)

There is no deterministic algorithm which always achieve consensus in the asynchronous model, with $f > 0$ crash failures.

IMPOSSIBILITY RESULT

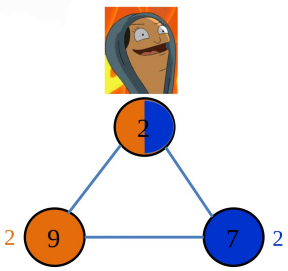
Theorem (1983)

There is a randomized algorithm achieving consensus in the asynchronous model if up to $f < n/2$ nodes crash. No consensus algorithm can tolerate $f \geq n/2$ many crash failures.

Theorem (1983)

There is a consensus algorithm for the synchronous model that tolerates any $f < n$ crash failures.

BYZ FAILURE



Theorem (1985)

There exists a randomized algorithm achieving consensus in the asynchronous model if up to $f < n/3$ nodes are byzantine.

IMPOSSIBILITY RESULT

Theorem (1989)

There is a deterministic algorithm achieving consensus in the synchronous model if up to $f < n/3$ nodes are byzantine.

Theorem (1980)

No consensus algorithm can work for $f \geq n/3$ byzantine nodes, even in the synchronous model.

Fair Consensus Problem

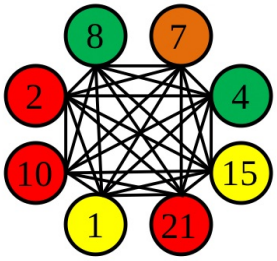
a set of n nodes

each node has:

unique ID

a color in {●●●…●}

an underlying communication graph G



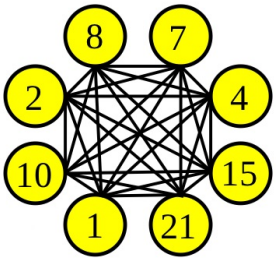
Goal: a *distributed protocol* guaranteeing

Termination (protocol eventually ends)

Agreement (monochromatic final configuration)

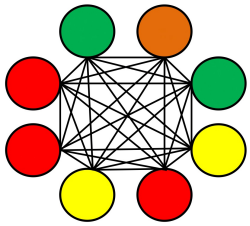
~~Validity~~ → Fairness

the probability that a color becomes the *winning color* is equal to the fraction of nodes initially supporting it



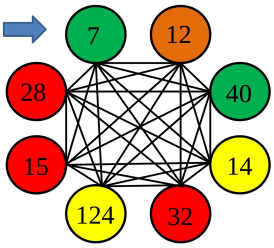
SIMPLE SOL: SCELTA DEL LEADER UNIFORME RANDOMIZZATA

- RANDOMIZED LEADER ELECTION

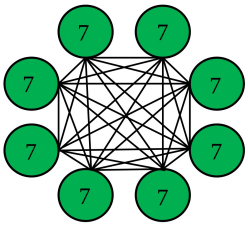


I NODI NON USANO I PROPRI IDs

the minimum k value is unique w.h.p.



SCELGONO $k_u = [0, n^3]$ UNIFORMEMENTE A CASO



- FIND-MIN PROTOC
- CONVERGENZA AL LEADER COLOR

BITCOIN LA PROB. DI DIVENTARE LEADER È PROPORZIONALE AL PROPRIO POTERE COMPUTAZIONALE

RATIONAL SELFISH FAILURE

A game-theoretic perspective

Each node is a selfish rational player

for each player/node u:

u's strategy: local algorithm used by u

u's payoff:

- ☺ if the winning color = u's color
- 😐 if the winning color ≠ u's color
- ☹ if the protocol fails

u's goal: to maximize its expected payoff

Rational Fair Consensus Problem

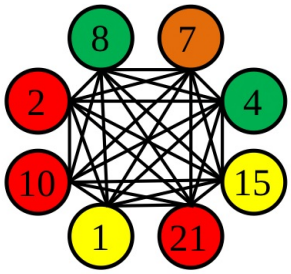
a set of n nodes

each node has:

unique ID

a color in {●●●●…●}

an underlying communication graph G



a protocol solves the Rational Fair Consensus if

it solves the Fair Consensus problem

&

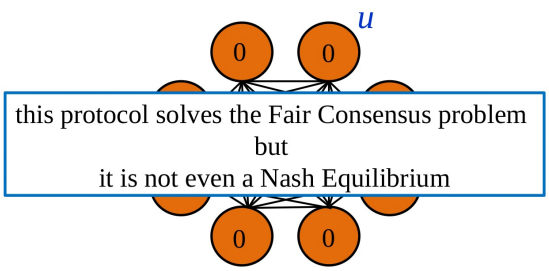
it is resilient to agent deviations

Resiliency to agent deviations

Def. 1. A protocol P is a **Nash Equilibrium** if, for any possible deviation of any agent u, u's expected payoff in the new protocol P' can only decrease.

Def. 2. A protocol P is a **t-strong Equilibrium** if, for any possible deviation of any coalition C of players of size at most t, there is a player in C whose expected payoff in the new protocol P' does not increase.

(the simple solution fails)



node u chooses a "random" value k_u =0

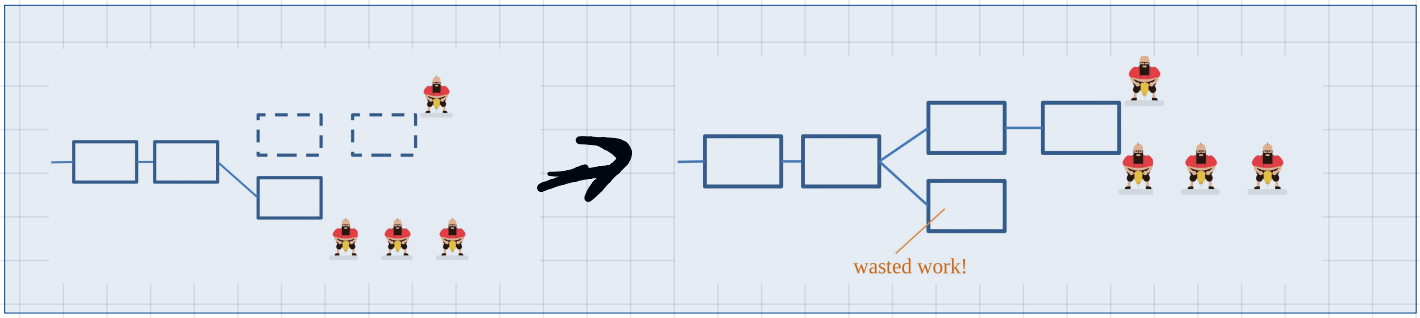
Bitcoin Mining Protocol:

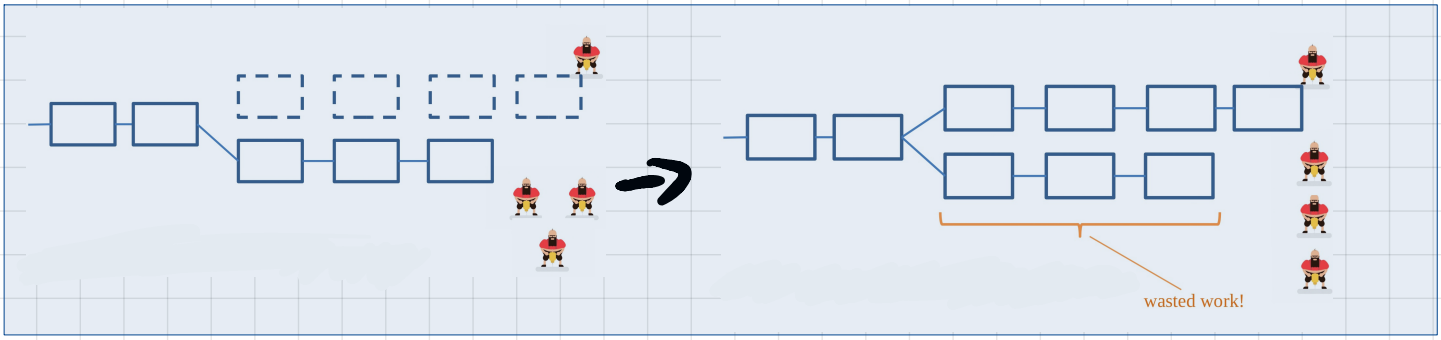
- work on the next block to be added to the longest chain
- announce the solved block as soon as you get it

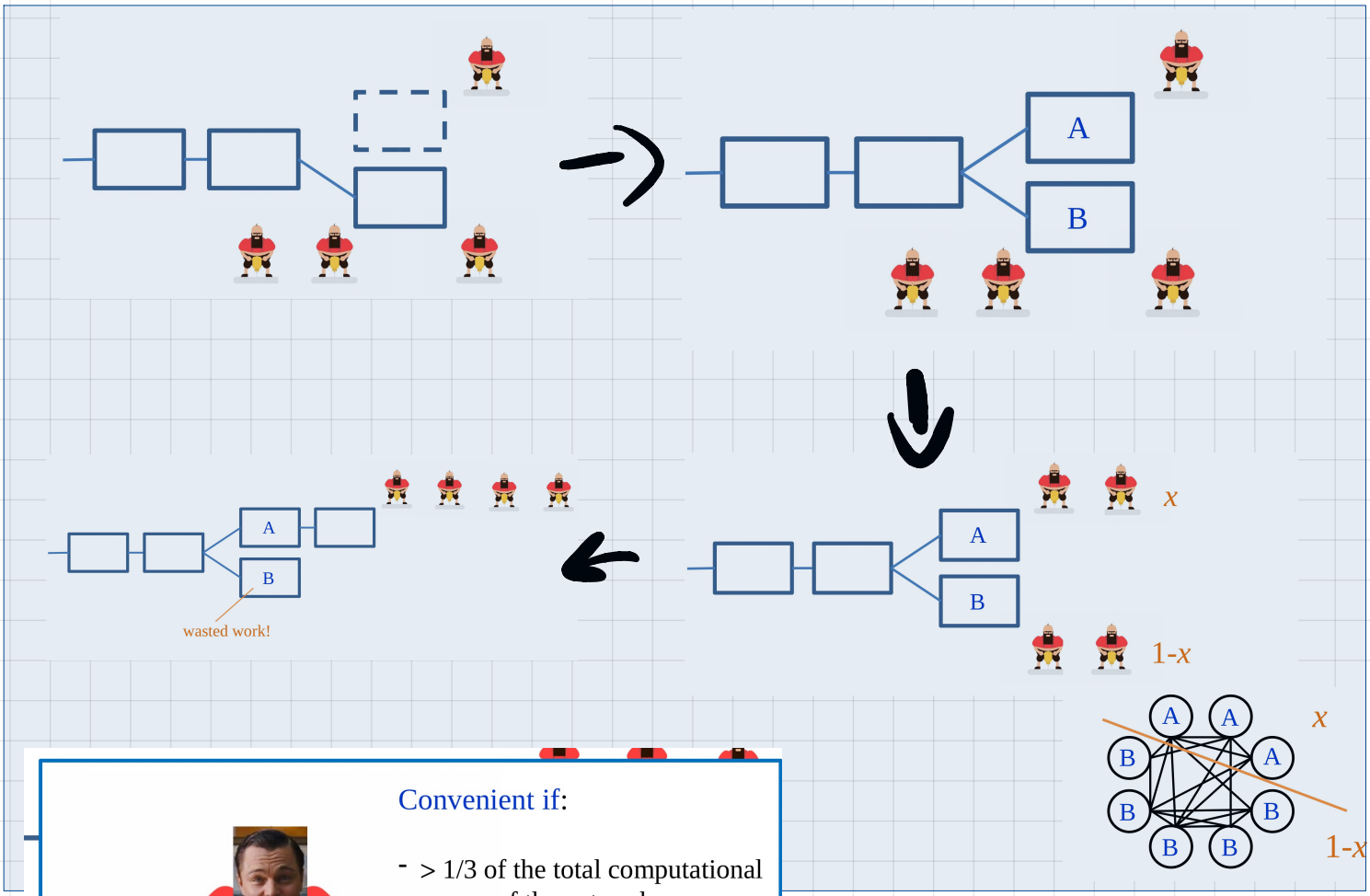
is it an equilibrium?

SELFISH MINING

IDEA: SELFISH DECIDE QUANDO ANNUNCIARE IL BLOCCO







Convenient if:

- > 1/3 of the total computational power of the network

OR

- x ≥ 1/2 & > 1/4 of the total computational power of the network