

Sisteme și algoritmi distribuiți

Curs 4

Problemă și ipoteze

Problema diseminării de jetoane: fie un set de jetoane (*tokens*) $M = \{v_1, v_2, \dots, v_k\}$, $|M| = k$, distribuite pe nodurile din rețea. Nodul i stochează în starea inițială doar token-ul propriu $v_i \in M$. Realizați distribuția jetoanelor astfel încât, în starea optimă orice nod i va deține toate jetoanele din M .

- Topologie reprezentată printr-un graf directat **tare conectat**.
- Fiecare nod are un **ID unic** id_i .
- Dimensiune token $\text{sizeof}(v_i) = \text{sizeof}(ID) = B$ biți.
- Notatii:
 - \mathcal{N}_i^- mulțimea vecinilor de intrare asociați nodului $i \in V$
 - \mathcal{N}_i^+ mulțimea vecinilor de ieșire asociați nodului $i \in V$
 - $\langle m_1, m_2 \rangle$ atașarea mesajului m_2 la m_1

Algoritmul FloodSet

Algoritm **FloodSet**():

M_i : - int id (id propriu)

- int v (token, inițial egal cu $x_i(0)$)

- int t , integer, inițial 0

Funcție transformare nod i ():

1. Fie U mulțimea mesajelor $\langle v_j, id_j \rangle$ primite de la \mathcal{N}_i^-
2. $M(t + 1) = M(t) \cup U$
3. Fie $V(t+1)$ mulțimea valorilor v_j din $M(t + 1)$
4. Fie $I(t+1)$ mulțimea id-urilor id_j din $M(t + 1)$
5. **If** ($I(t+1) == I(t)$): **STOP**;
6. **Else**: send($M(t + 1)$, \mathcal{N}_i^+)
7. $t := t + 1$

Teorema [Kuhn et al.]. În algoritmul FloodSet, $M_i(t) = M$ după $O(diam)$ iterații.

1. FloodSet folosește mesaje $O(n B)$.
2. FloodSet necesită memorie $O(n B)$.
3. FloodSet nu necesită cunoașterea lui n sau $diam$.
4. Analiza complexității este similară cu cea din cazul **Flooding_gen**(max()).
5. FloodSet este un tipar algorithmic care se poate aplica pentru calcularea oricărei funcții.

Adaptare FloodSet pentru calcul distribuit

Aplicație: Calculați distribuit valoarea funcției f în x , i.e. $x_i^* = f(x_1, x_2, \dots, x_n)$.

- Considerăm $v_i := x_i$, i.e. $M = \{x_1, x_2, \dots, x_n\}$, $|M| = n$.
- Nodul i pornește din $x_i(0) := v_i$, $\forall i \in V$, converge către $x_i^* = f(x_1, \dots, x_n)$.

Păstrăm ipotezele anterioare:

- Topologie reprezentată printr-un graf directat **tare conectat**.
- Fiecare nod are un **ID unic** id_i .
- Dimensiune token $\text{sizeof}(v_i) = \text{sizeof}(ID) = B$ biți.

Adaptare FloodSet pentru alte funcții

Algoritm **FloodSet**($f, x(0)$):

M_i : - int id (id propriu)

- int v (token, inițial egal cu x_i)

- funcție obiectiv $f()$

- int t , integer, inițial 0

Funcție transformare nod i ():

1. Fie U mulțimea mesajelor $\langle v_j, id_j \rangle$ primite de la \mathcal{N}_i^-

2. $M(t+1) = M(t) \cup U$

3. Fie $V(t+1)$ mulțimea valorilor v_j din $M(t+1)$

4. Fie $I(t+1)$ mulțimea id-urilor id_j din $M(t+1)$

5. **If** ($I(t+1) == I(t)$):

1. Return $f(V(t))$

6. **Else**: send($M(t+1)$, \mathcal{N}_i^+)

7. $t := t + 1$

Teorema [Kuhn et al.]. Algoritmul **FloodSet**($f, x(0)$) returnează valoarea lui $f(x(0))$ după $O(diam)$ iterații.

1. FloodSet folosește mesaje $O(n B)$.
2. FloodSet necesită memorie $O(n B)$.
3. FloodSet nu necesită cunoașterea lui n sau $diam$;
4. Analiza complexității este similară cu cea din cazul **Flooding_gen**(max()).

Ipoteze și premise

- Topologie reprezentată printr-un graf directat **tare conectat**.
- ~~Fiecare nod are un **ID unic** id_i .~~ (Rețele anonime)
- Fiecare nod are un token $v_i \subset M$, un M mulțime total ordonată.
- Dimensiune token $\text{sizeof}(v_i) = \text{sizeof}(ID) = B$ biți.
- Urmărim algoritmi cu necesar de memorie/mesaj $< O(nB)$.
- Teorema de imposibilitate pentru rețele anonime.

Teorema de imposibilitate pentru rețele anonime

Ipoteze model distribuit:

- Noduri identice
- Rețea anonimă
- Determinism
- Memorie locală limitată (e.g. creștere slabă funcție de gradul nodului)
- Absența informației globale (P_i cunoaște doar vecinii de intrare)
- Topologie statică

Teorema de imposibilitate pentru rețele anonime

O funcție $f: R^n \rightarrow R^n$ este *independentă de ordine și multiplicitate* dacă valoarea ei este complet determinată de mulțimea valorilor care apar în vectorul $x \in R^n$ (indiferent de ordinea și numărul de apariții), *i.e.*

$$\exists g \text{ a.î. } f(x) = g(\{v: \exists i : v = x_i\}).$$

Exemple:

- $f(x) = \max(x_1, \dots, x_n)$
- $f(x) = \min(x_1, \dots, x_n)$
- Contraexemplu: $f(x) = \frac{1}{n} \sum_{i=1}^n x_i$

Teorema de imposibilitate pentru rețele anonime

Teorema de imposibilitate [Hendricx&Tsitsiklis]. Dacă o funcție f este calculabilă de modelul distribuit specificat anterior, atunci f este *independentă de ordine și multiplicitate*.

Concluzii:

- Algoritmii pentru Alegere Lider (e.g. Flooding) nu necesită informație globală pentru a rezolva problema AL
- Pentru a calcula funcții *dependente de ordine sau multiplicitate*, trebuie eliminată cel puțin o ipoteză a modelului.

Recapitulare alg. Flooding(max())

Algoritm **Flooding_Gen**(max()):

M_i : - int v (token, inițial egal cu $x(0)$)
- int max_v (var auxiliară), inițial v
- $status \in \{\text{lider, non-lider}\}$, inițial non-lider
- int t , integer, inițial 0
- int $diam$ (diametru graf)

Funcție transformare nod i ():

1. $t := t + 1$
2. Fie U mulțimea token-urilor primite de la \mathcal{N}_i^-
3. $max_v := \max(\{max_v\} \cup U)$
4. **If** ($t == diam$):
 1. **If** ($max_v == v$): status = leader;
 2. **Else**: status = non-leader;
5. **Else**: send(v , \mathcal{N}_i^+)

Teorema [Lynch]. În algoritmul Flooding, nodul cu indicele i_{max} este lider, restul nodurilor non-lider, după $diam$ iterații.

Complexitate. Complexitatea timp este $diam$ iterații până la anunțarea unui lider, iar complexitatea mesaj este $diam \cdot |E|$. Prin $|E|$ înțelegem numărul de muchii directate din graf.

Recapitulare alg. Flooding(max())

Algoritm **Flooding_Gen**(max()):

M_i : - int v (token, inițial egal cu $x(0)$)
- int max_v (var auxiliară), inițial v
- $status \in \{\text{lider, non-lider}\}$, inițial non-lider
- int t , integer, inițial 0
- int $diam$ (diametru graf)

Funcție transformare nod i ():

1. $t := t + 1$
2. Fie U mulțimea token-urilor primite de la \mathcal{N}_i^-
3. $max_v := \max(\{max_v\} \cup U)$
4. **If** ($t == diam$):
 1. **If** ($max_v == v$): status = leader;
 2. **Else**: status = non-leader;
5. **Else**: send(v , \mathcal{N}_i^+)

Starea $x_i(t)$ este tokenul maxim max_v la momentul t .

$$\mathcal{N}_i = \{x_{i_1}(t), \dots, x_{i_{|N_i|}}(t)\}$$
$$x_i(t+1) = \max(x_i(t), x_{i_1}(t), \dots, x_{i_{|N_i|}}(t))$$

1. Flooding(max()) folosește mesaje $O(B)$
2. Flooding(max()) necesită memorie $O(B)$
3. Flooding(max()) necesită cunoașterea unei estimări a lui $diam$ (pt criteriu de oprire)

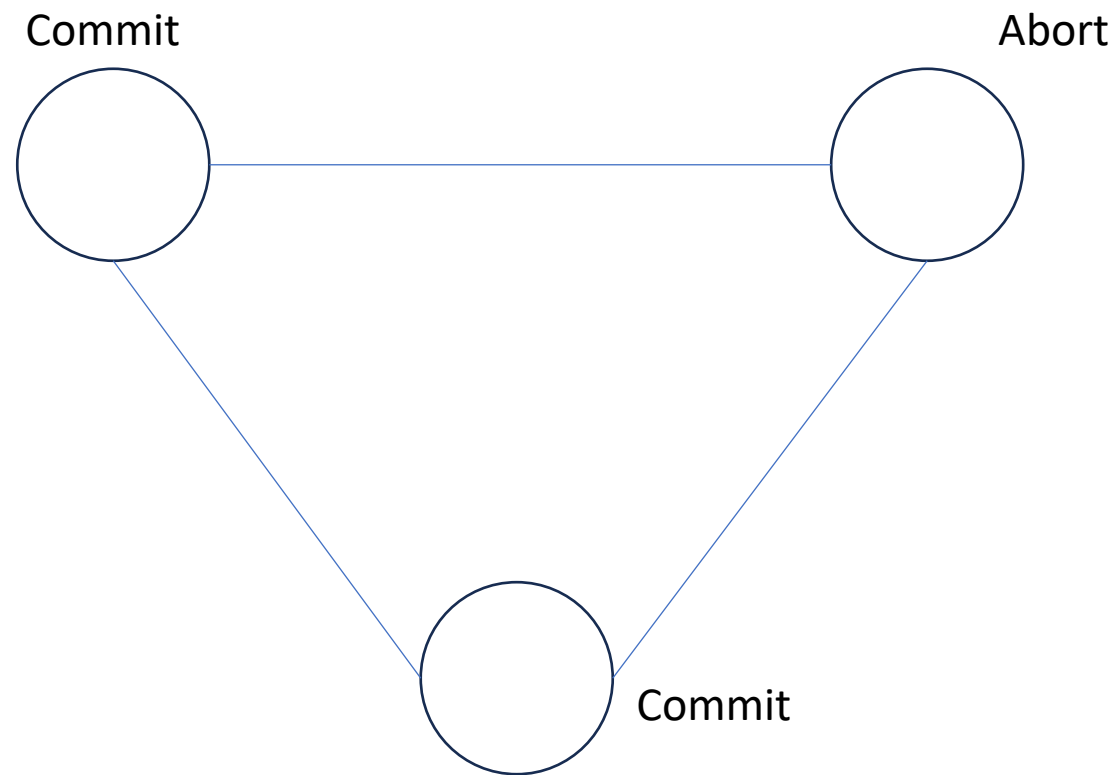
Problemă

Fie $a \in R^n$. Notăm $a_{[1]} \leq a_{[2]} \leq \dots \leq a_{[n]}$. Adaptați algoritmul Flooding pentru a calcula $a_{[n]} + a_{[n-1]}$.

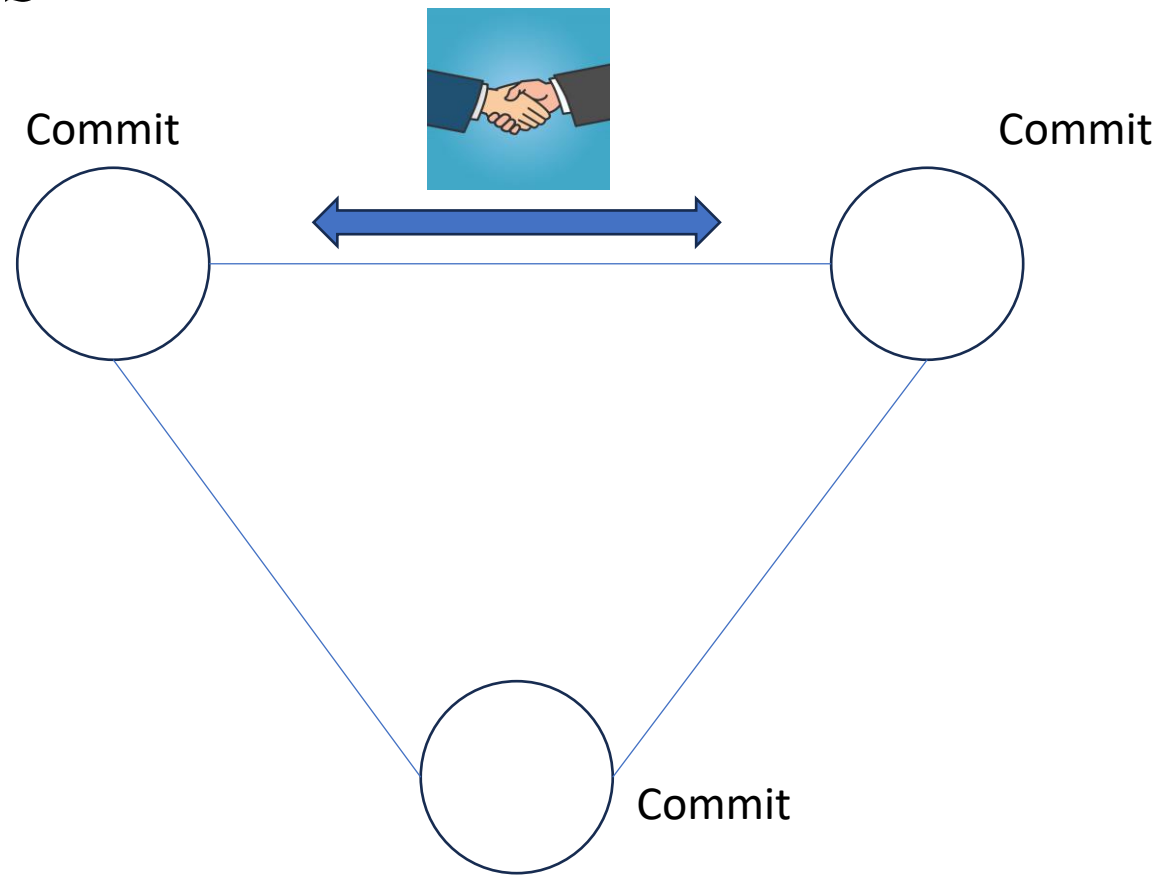
Rezolvare: la tablă.

Consens

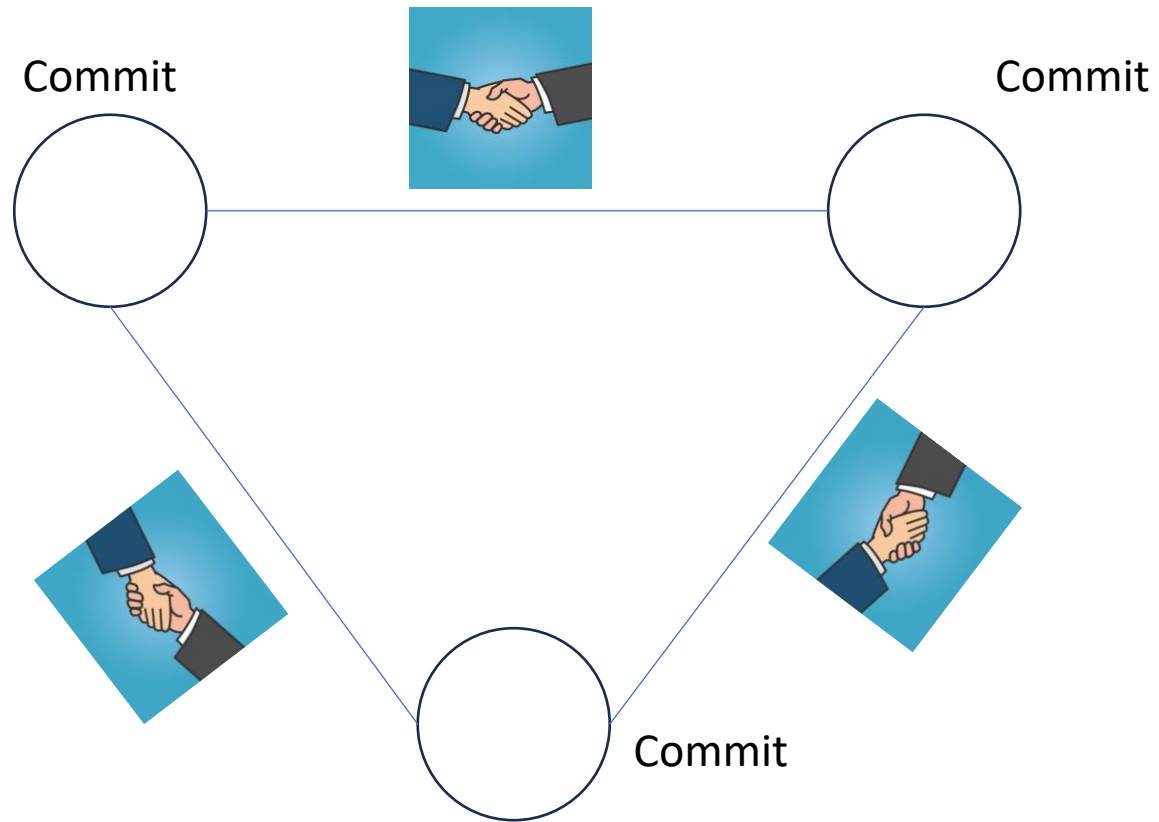
Consens



Consens

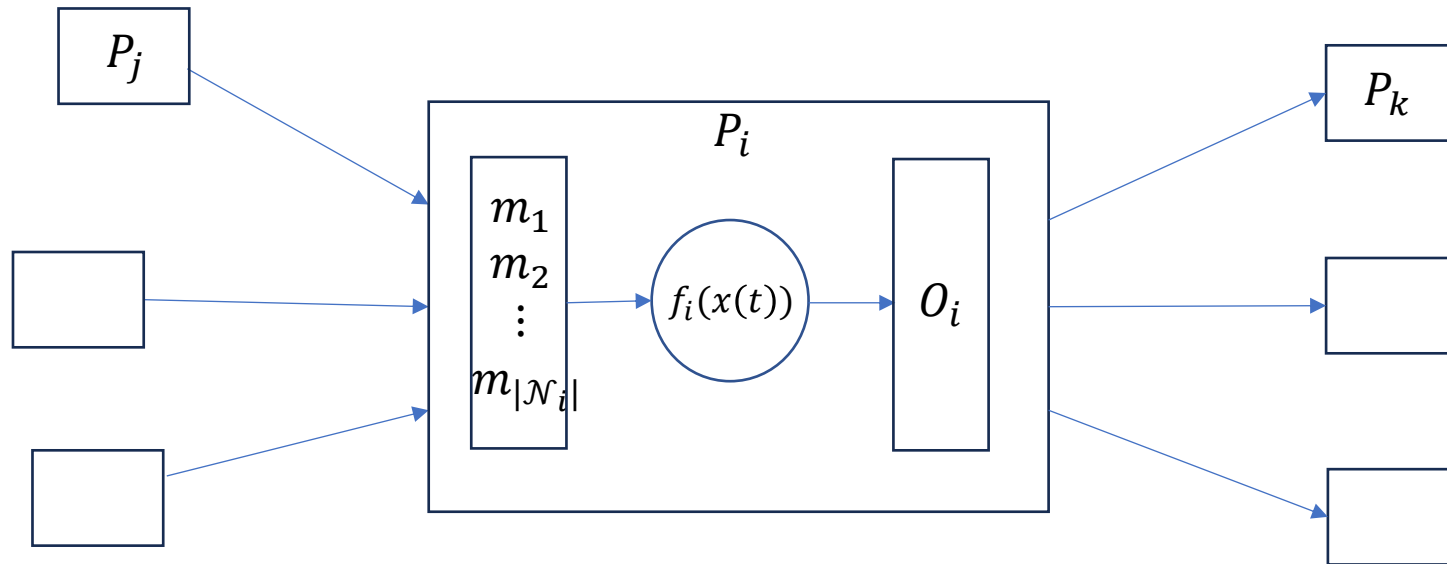


Consens



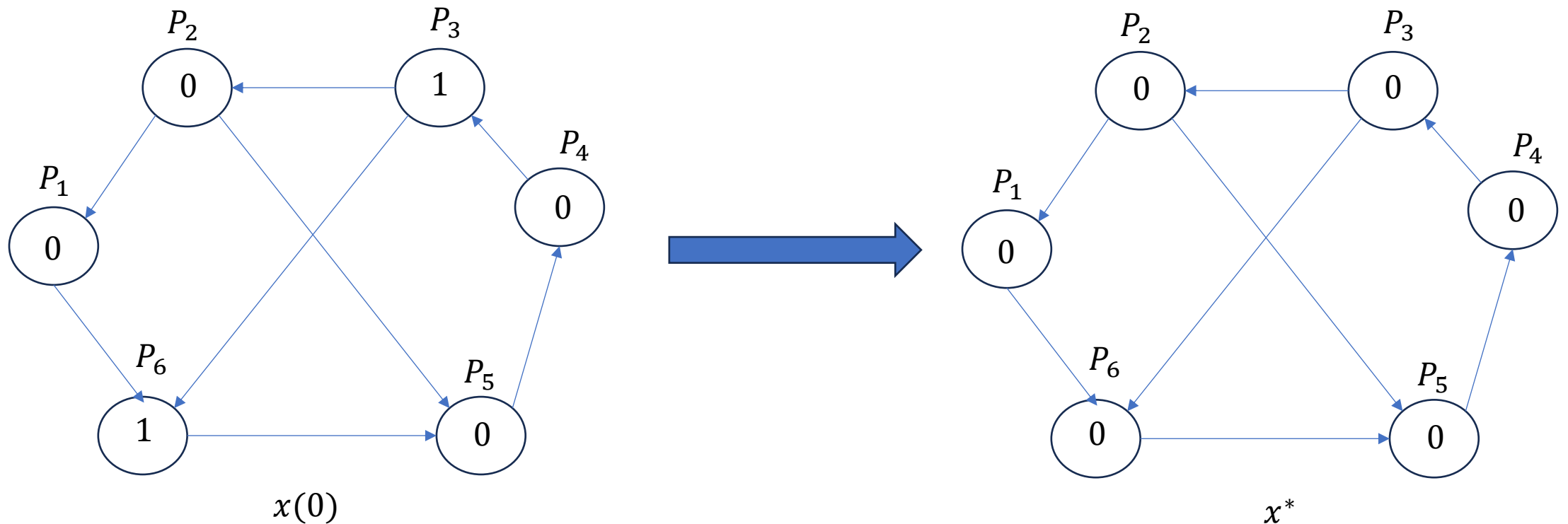
Consensus: definiție

Starea (valoarea) uniformă a nodurilor unui sistem distribuit.



Consens: definiție

Starea (valoarea) uniformă a nodurilor unui sistem distribuit.



Consens: definiție

Starea (valoarea) uniformă a nodurilor unui sistem distribuit.

Condiția de acord: Nu există două procese care decid valori diferite.

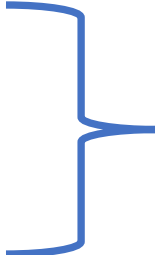
Condiția de validitate: Dacă valoarea inițială a proceselor este v , atunci consensul se atinge cu valoarea uniformă v .

Condiția de terminare (algorithm): Într-un algorithm de consens, orice nod din sistem va decide eventual la un moment de timp.

Adesea se reduce la calcularea distribuită a valorii unei funcții de consens în starea inițială a sistemului.

Consensus

Example:

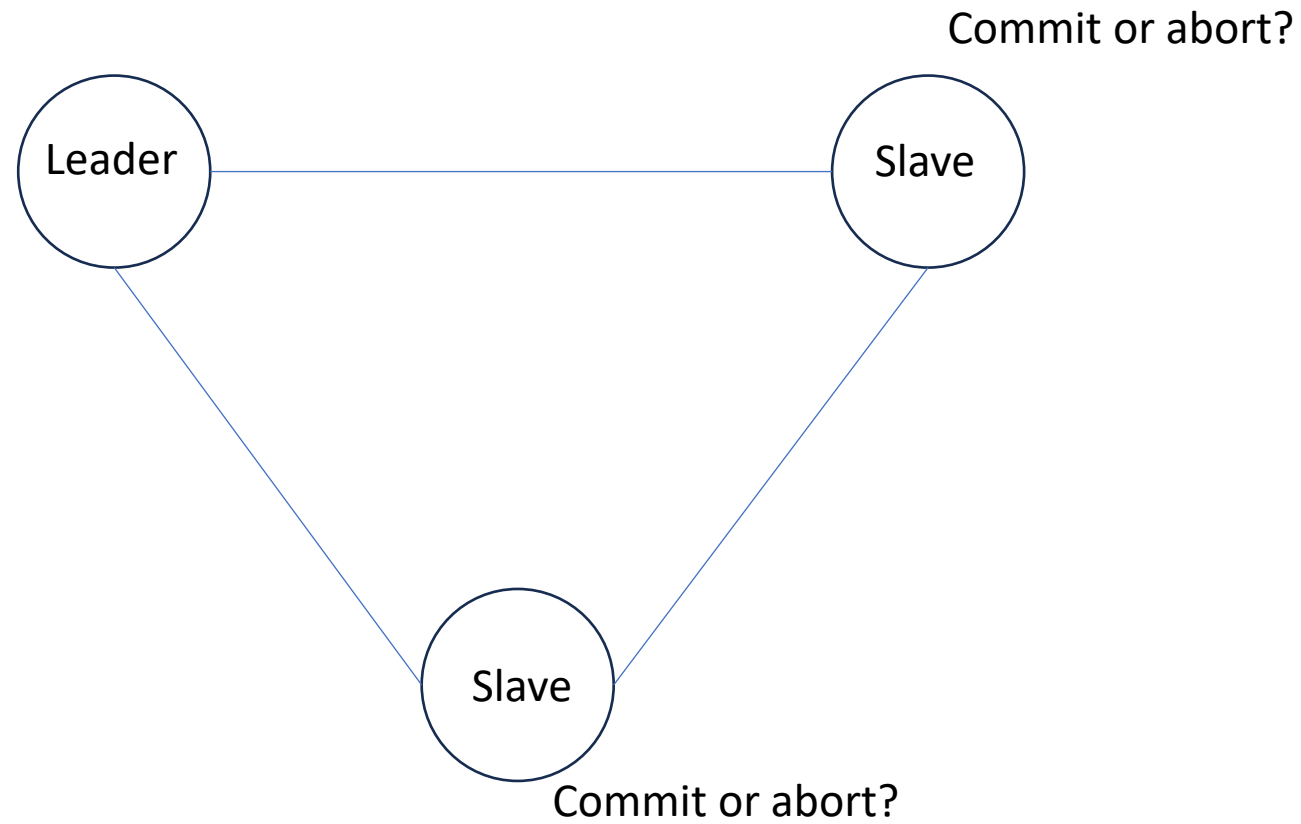
- Majoritar $x_i^* = Maj(x(0)) = \begin{cases} 1, \text{dacă } |\{i | x_i(0) = 1\}| \geq \frac{n}{2} + 1 \\ 0, \text{dacă } |\{i | x_i(0) = 1\}| < \frac{n}{2} + 1 \end{cases}$
 - Medie (aritmetică) $x_i^* = \frac{1}{n} \sum_{i=1}^n x_i(0)$
 - Mediană $x_i^* = x_{[\frac{n}{2}]}(0)$
 - Max-consensus $x_i^* = \max_{1 \leq i \leq n} \{x_i(0)\}$
 - Min-consensus $x_i^* = \min_{1 \leq i \leq n} \{x_i(0)\}$
- 
- Funcții independente de ordine și multiplicitate

Consens distribuit: aplicații

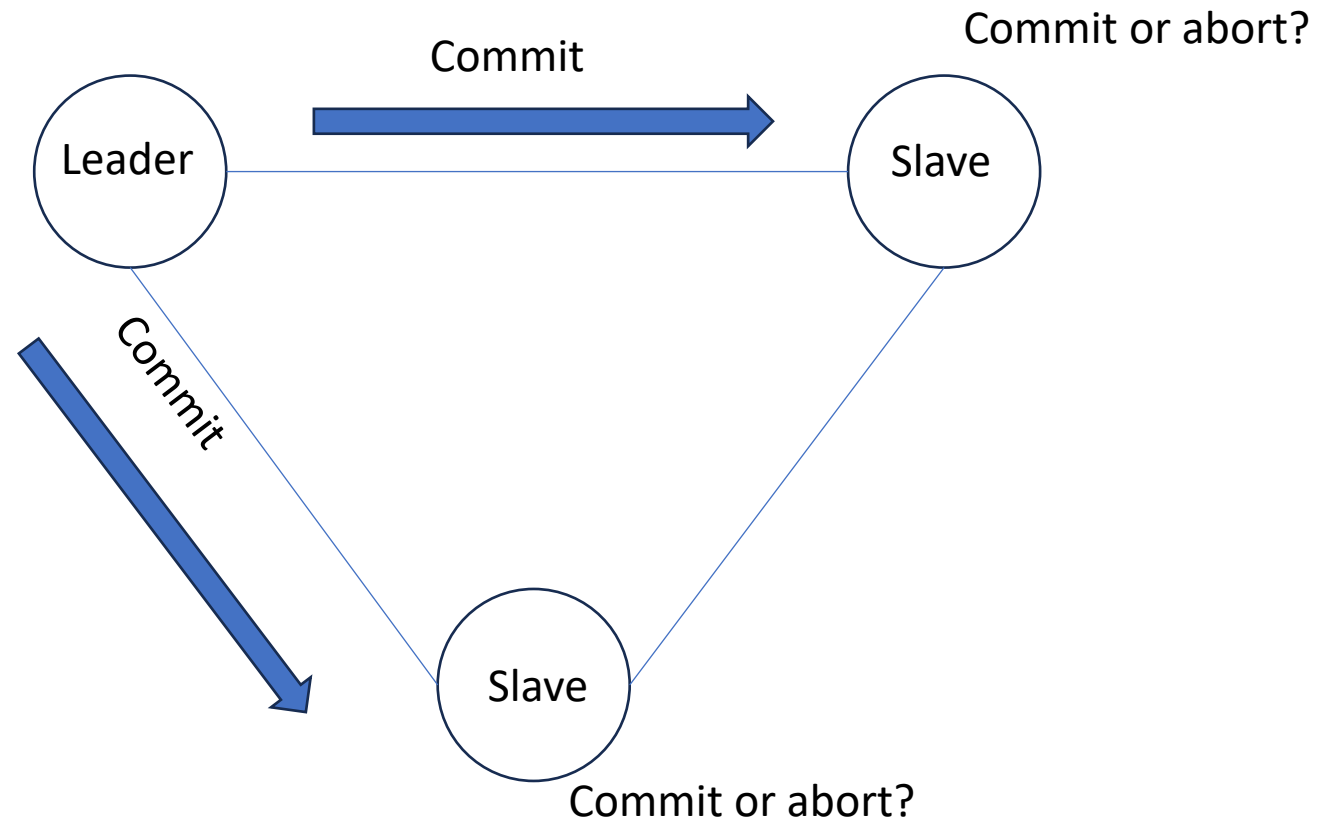
Calculul consensului este necesar în operații de nivel înalt:

- Elecția liderului (max-consens)
- Sincronizare ceasuri (medie dinamică, max-consens dinamic)
- Asigurare consistență baze de date (majoritar)
- COMMIT distribuit (majoritar)
- Localizare distribuită în rețele de senzori (medie)
- Formarea de grupuri în rețele de agenți (medie dinamică)

Consensus centralizat



Consensus centralizat



Consens centralizat

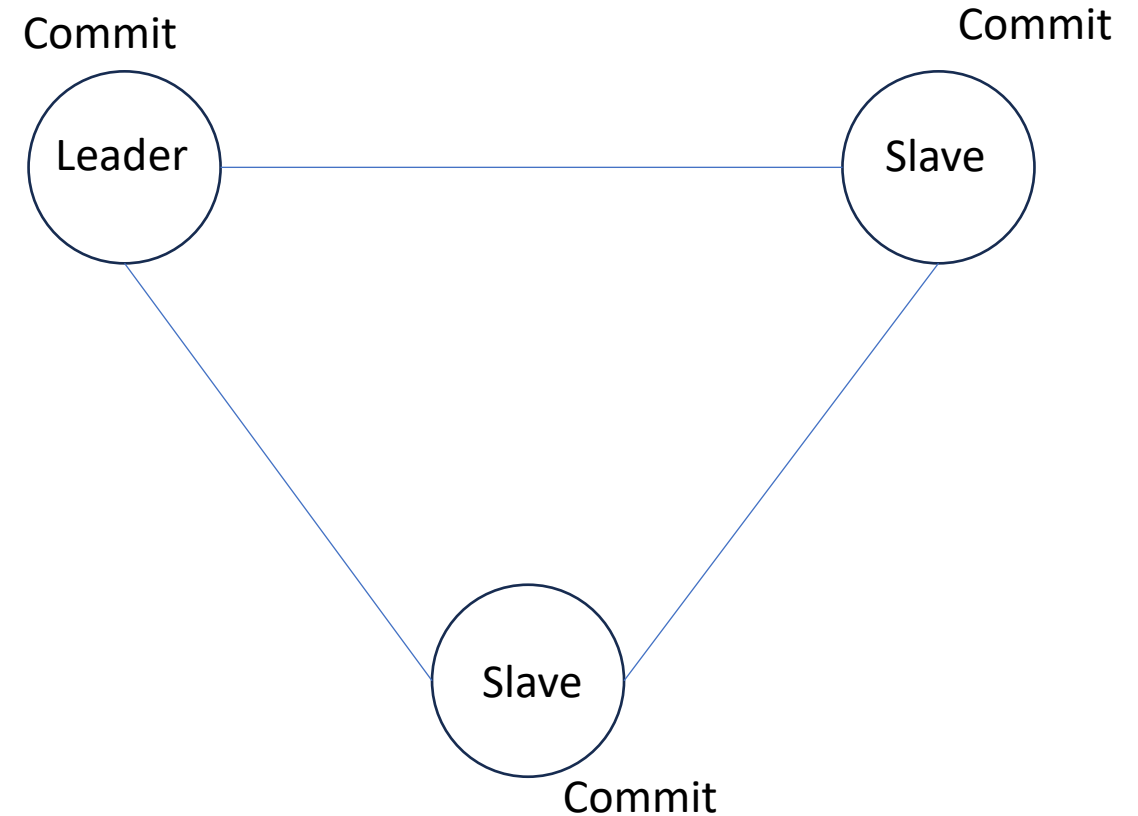
În context sincron fără defecte, asigurarea consensului centralizat se realizează printr-o simplă difuzarea de mesaje.

Dificultatea rămâne selecția preliminară a liderului, care se realizează folosind algoritmi sincroni AL (vezi cursul trecut).

Consens binar majoritar

- stare lider x_l

1. $\text{buf} = \text{Gather}(G);$
2.
$$x_l = \begin{cases} 1, & \text{dacă } |\{buf_i = 1\}| \geq \frac{n}{2} + 1 \\ 0, & \text{dacă } |\{buf_i = 1\}| < \frac{n}{2} + 1 \end{cases}$$
3. $\text{Broadcast}(x_l);$



Consens centralizat

Consens binar majoritar (centralizat)

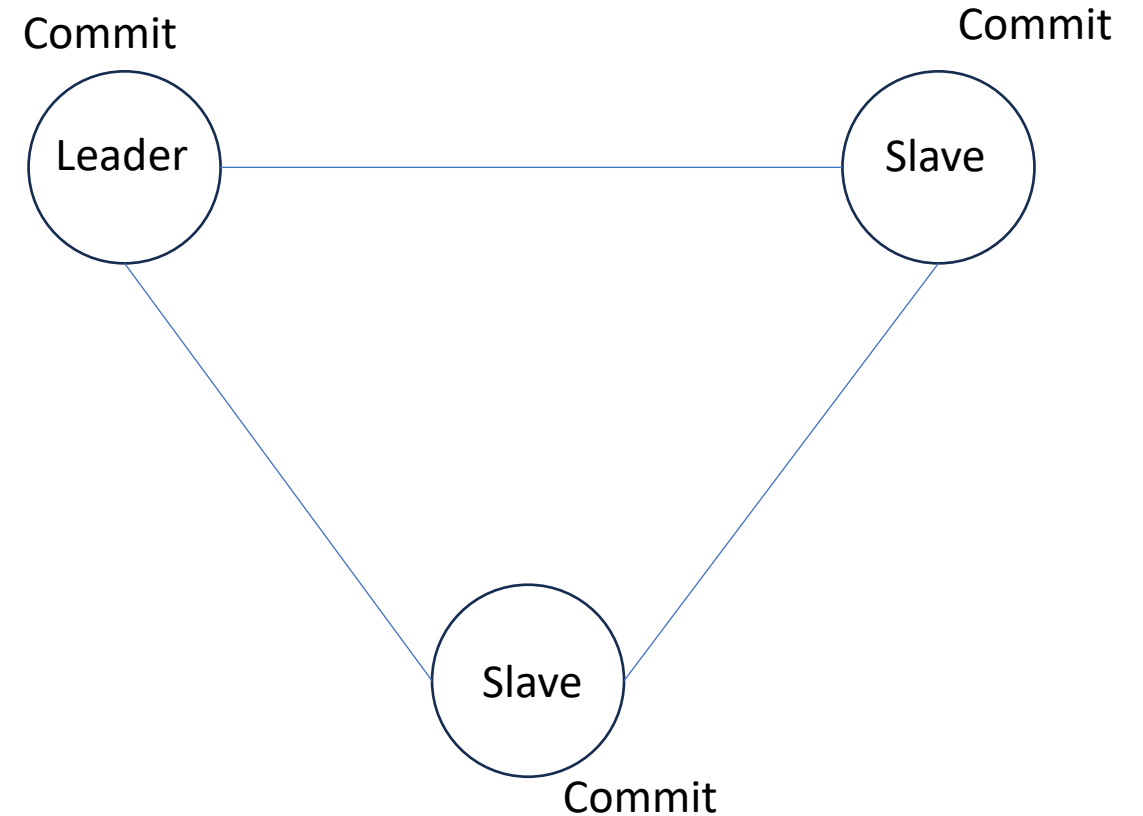
- stare lider $x_l \in \{0,1\}$

1. $\text{buf} = \text{Gather}(G)$;

2.
$$x_l = \begin{cases} 1, & \text{dacă } |\{buf_i = 1\}| \geq \left\lceil \frac{n}{2} \right\rceil \\ 0, & \text{dacă } |\{buf_i = 1\}| < \left\lceil \frac{n}{2} \right\rceil \end{cases}$$

3. $\text{Broadcast}(x_l)$;

- Slave: stochează 1 bit cu decizia curentă.
- Dacă avem perturbații pe noduri sau pe legături, schema de mai sus nu funcționează (rezultat posibil greșit).
- De asemenea, un lider defect impune reluarea procedurii de AL distribuit.



Consens distribuit

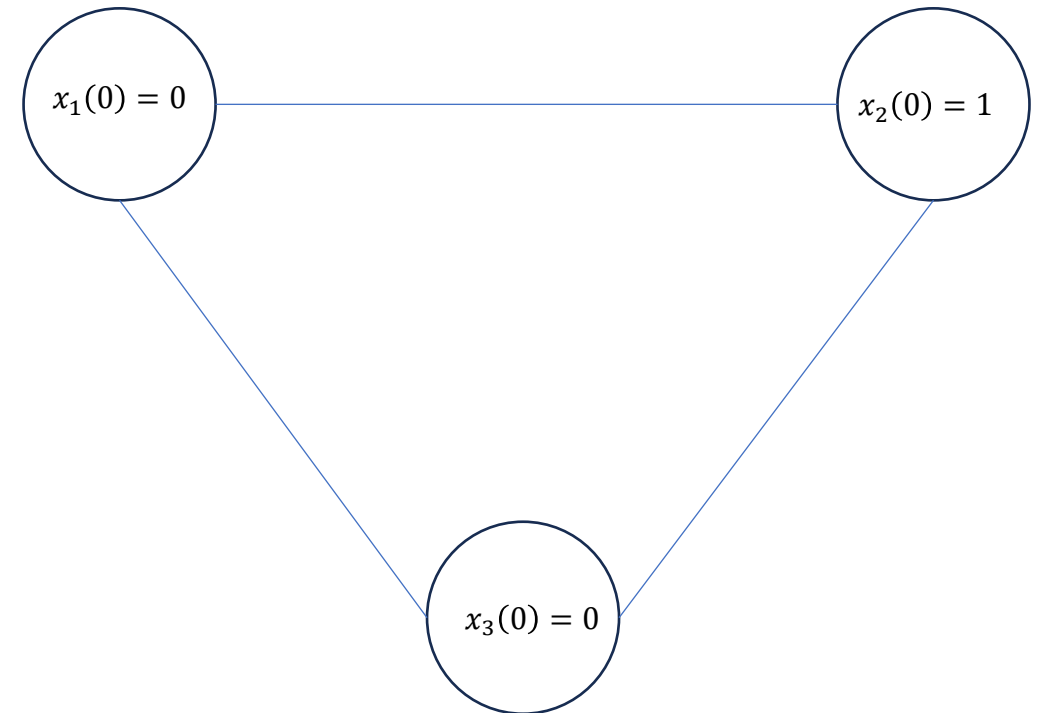
Cf. Teoremei de imposibilitate **Hendricx&Tsitsiklis**, dacă funcția de consens **nu este** independentă de ordine și multiplicitate atunci consensul este imposibil de atins fără cel puțin un atribut precum:

- informație globală *e. g.* n , $diam(G)$, G
- capacitate locală de stocare mare $B > \deg(P_i)$
- o distribuție de identificatori

Consens binar majoritar (distribuit)

- stare lider $x_i \in \{0,1\}$

1. $buf_i = \text{Gather}(\mathcal{N}_i)$;
2.
$$x_i = \begin{cases} 1, & \text{dacă } |\{buf_i[j] = 1\}| \geq \left\lceil \frac{|\mathcal{N}_i|}{2} \right\rceil \\ 0, & \text{dacă } |\{buf_i[j] = 1\}| < \left\lceil \frac{|\mathcal{N}_i|}{2} \right\rceil \end{cases}$$
3. $\text{Broadcast}(x_i, \mathcal{N}_i)$;



Consens distribuit

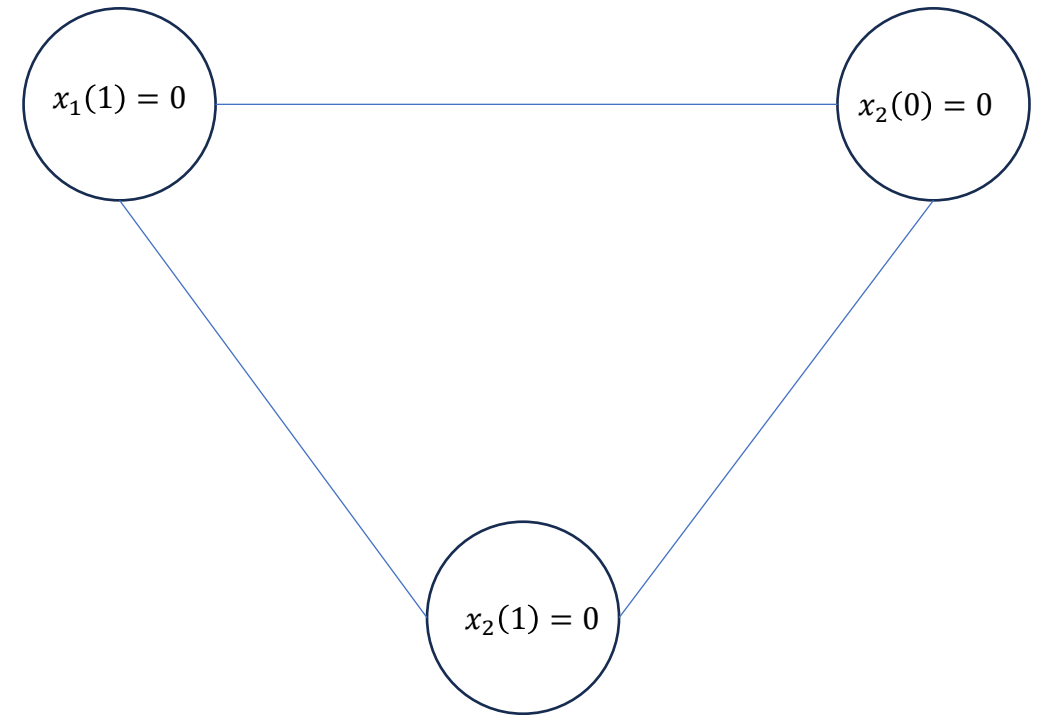
Consens binar majoritar (distribuit)

- stare lider $x_i \in \{0,1\}$

1. $buf_i = \text{Gather}(\mathcal{N}_i);$
2. $x_i = \begin{cases} 1, & \text{dacă } |\{buf_i[j] = 1\}| \geq \left\lceil \frac{|\mathcal{N}_i|}{2} \right\rceil \\ 0, & \text{dacă } |\{buf_i[j] = 1\}| < \left\lceil \frac{|\mathcal{N}_i|}{2} \right\rceil \end{cases}$
3. $\text{Broadcast}(x_i, \mathcal{N}_i);$

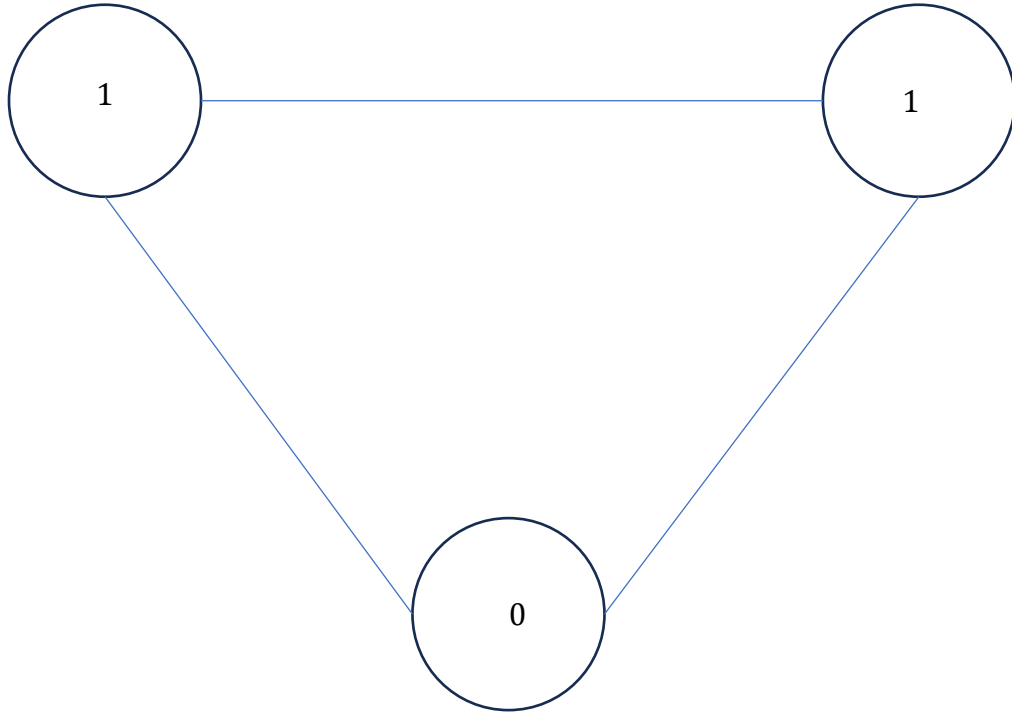
Actualizări locale bazate pe calculul „majorității” valorilor provenite de la vecini.

Dificultate majoră: natura binară a stărilor locale $x_i \in \{0,1\}$, $B = 1$.

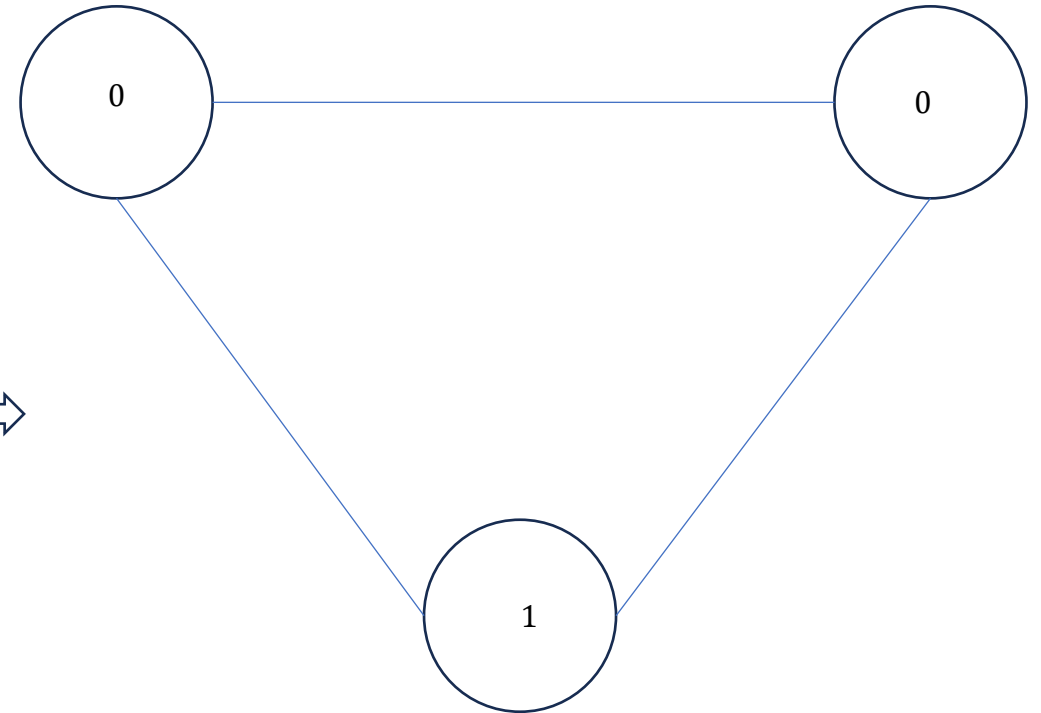


Consensus distribuit

$x(0)$

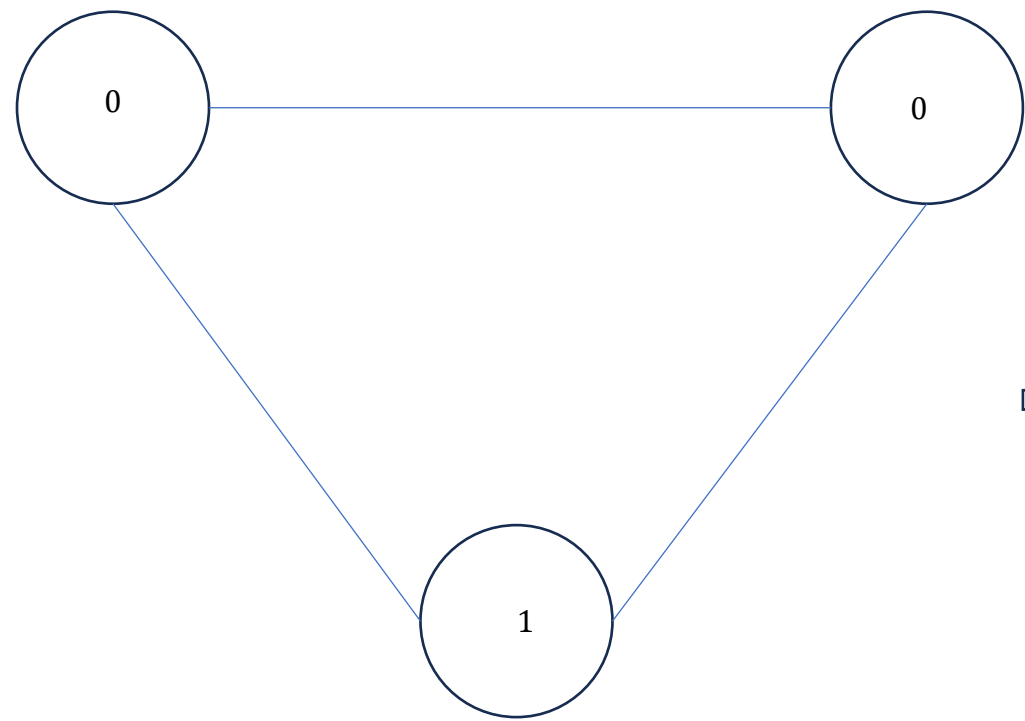


$x(1)$

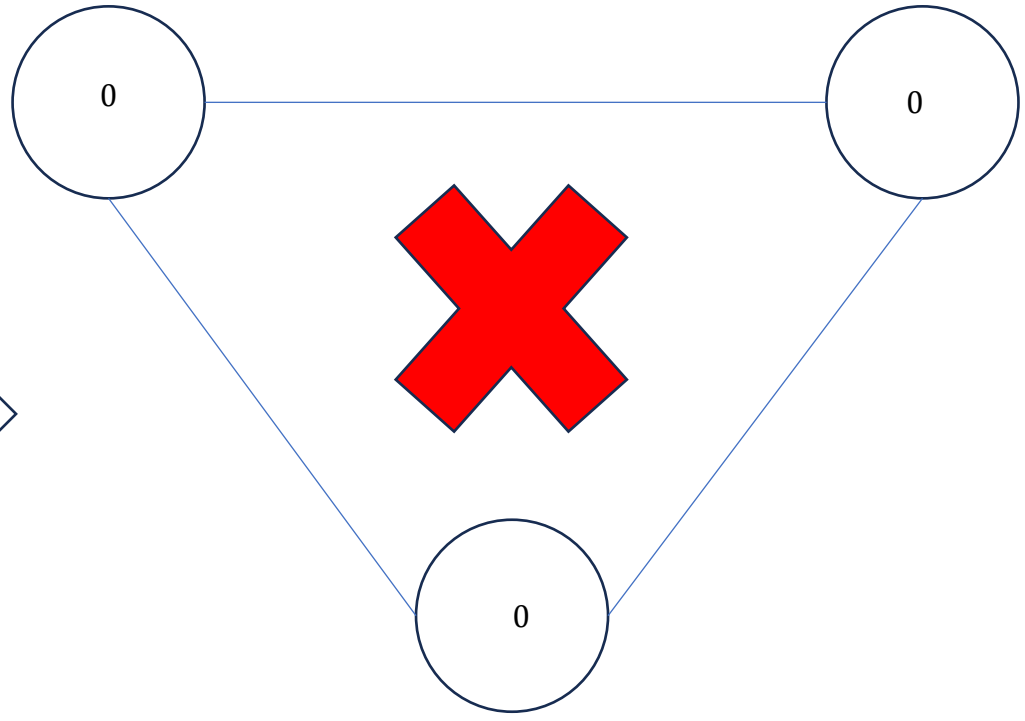


Consensus distribuit

$x(1)$



$x(2)$



Consens distribuit

Teoremă de imposibilitate [Land & Belew]. Fie sistemul $(\{x(t)\}_{t \geq 0}, \mathcal{G})$ cu n noduri și stări binare $x(t) \in \{0,1\}^n$. Nu există un algoritm determinist, sincron, distribuit care rezolvă exact *problema de consens binar majoritar* (pentru oricare \mathcal{G}).

Concluzie: Numărul (natura) stărilor per nod este un factor important în rezolvarea distribuită a problemelor centralizate.

Consens distribuit

Teoremă de imposibilitate [Land & Belew]. Fie sistemul $(\{x(t)\}_{t \geq 0}, \mathcal{G})$ cu n noduri și stări binare $x(t) \in \{0,1\}^n$. Nu există un algoritm determinist, sincron, distribuit care rezolvă exact *problema de consens binar majoritar* (pentru oricare \mathcal{G}).

Cum depășim teorema de imposibilitate?

- Stări multiple (e.g. reale, nu binare), $B > 1$
- Automate probabilistice
- Automate asincrone

Algoritm FloodSet pentru consens

Păstrăm ipotezele anterioare:

- Topologie reprezentată printr-un graf directat **tare conectat**.
- Fiecare nod are un **ID unic** id_i .
- Dimensiune token $\text{sizeof}(v_i) = \text{sizeof}(ID) = B$ biți.

Problemă: Calculați distribuit valoarea funcției f de consens în $x(0)$, astfel încât $x_i^* = f(x(0))$.

- Considerăm $v_i := x_i$, *i. e.* $M = \{x_1, x_2, \dots, x_n\}$, $|M| = n$.
- Nodul i pornește din $x_i(0) := v_i$, $\forall i \in V$, converge către $x_i^* = f(x(0))$.
- Problema este o aplicație a soluției problemei de diseminare de jetoane.

Algoritm FloodSet pentru consens

Algoritm **FloodSet**($f()$):

M_i : - int id (id propriu)

- int v (token, inițial egal cu x_i)

- funcție obiectiv $f()$

- int t , integer, inițial 0

Funcție transformare nod $i()$:

1. Fie U mulțimea mesajelor $\langle v_j, id_j \rangle$ primite de la \mathcal{N}_i^-

2. $M(t+1) = M(t) \cup U$

3. Fie $V(t+1)$ mulțimea valorilor v_j din $M(t+1)$

4. Fie $I(t+1)$ mulțimea id-urilor id_j din $M(t+1)$

5. **If** ($I(t+1) == I(t)$):

1. **Return** $f(M(t))$

6. **Else**: send($M(t+1)$, \mathcal{N}_i^+)

7. $t := t + 1$

- Reducem operația de consens static la calculul unei funcții de consens $f(x(0))$
- Dezavantaje:
 1. FloodSet folosește mesaje $O(n B)$
 2. FloodSet necesită memorie $O(n B)$
- În general urmărim ca dimensiunea mesajelor/memoriei să fie o funcție slab crescătoare de numărul de noduri (e.g. $\log(n)$, $n^{\frac{1}{p}}$)
- Consens majoritar: considerarea de stări reale ne conduce la algoritmi eficienți.

Consens majoritar

Fie $v \in R^n$, atunci

$$Maj(v) = \begin{cases} 1, & \text{dacă } |\{i | v_i = 1\}| \geq \frac{n}{2} + 1 \\ 0, & \text{dacă } |\{i | v_i = 1\}| < \frac{n}{2} + 1 \end{cases}$$

se rescrie notând $\bar{v} = \frac{1}{n} \sum_{i=1}^n v_i$

$$Maj(v) = \begin{cases} 1, & \text{dacă } \bar{v} \geq \frac{1}{2} \\ 0, & \text{dacă } \bar{v} < \frac{1}{2} \end{cases}$$

Consens majoritar

Păstrăm ipotezele anterioare:

- Topologie reprezentată printr-un graf directat **tare conectat**.
- Dimensiune token $\text{sizeof}(v_i) = \text{sizeof}(ID) = B$ biți.

Problemă: Calculați distribuit valoarea funcției $f() = \text{Maj}()$ de consens în $x(0)$, astfel încât $x_i^* = f(x(0))$.

- Considerăm $v_i := x_i$, *i. e.* $M = \{x_1, x_2, \dots, x_n\}$, $|M| = n$.
- Nodul i pornește din $x_i(0) := v_i$, $\forall i \in V$, converge către $x_i^* = f(x(0))$.
- Observație: $\text{Maj}(v) = \frac{1}{2} \left(1 + \text{sgn} \left(\text{Mean}(v) - \frac{1}{2} \right) \right)$

Algoritm Flooding pentru consens majoritar

Algoritm **Flooding**(Maj()):

M_i : - int v (token)
- int d (grad intrare), integer
- int t , integer, inițial 0

- Inițial: $x_j(0) = v_j \in R$

- Iterație locală: $x_i(t+1) = \frac{1}{d_i+1} \left(x_i(t) + \sum_{j \in \mathcal{N}_i^-} x_j(t) \right), \quad \forall i$

- Analiza complexității timp pe scurt: la tablă!

Funcție transformare nod i ():

1. Fie U mulțimea mesajelor $v_j = x_j(t)$ primite de la \mathcal{N}_i^-
2. $x(t+1) = \frac{1}{d+1} \left(x_i(t) + \sum_{j \in \mathcal{N}_i^-} x_j(t) \right)$
3. **If** (criteriu_oprire):
 1. **Return** $\frac{1}{2} \left(1 + \text{sgn} \left(x(t) - \frac{1}{2} \right) \right)$
4. **Else**: send($x(t+1)$, \mathcal{N}_i^+)
5. $t := t+1$