Sisteme și algoritmi distribuiți
Curs 5

Cuprins

- Consens – convergență (continuare curs 4)
- Defecte
- Algoritmi robuști la defect crash
- Algoritmi robuști la defect bizantin

Algoritm Flooding cu ponderi uniforme

Algoritm **Flooding**(Maj, v):

Mem_i: - int x_i (token) , inițial v_i
- int d (grad intrare), integer
- int *t*, integer, inițial 0

Funcție transformare nod i ():

1. Fie U mulțimea mesajelor $v_j = x_j(t)$ primite de la $\mathcal{N}_i^-$
2. $x_i(t+1) = \frac{1}{d+1}\left(x_i(t) + \sum_{j \in \mathcal{N}_i^-} x_j(t)\right)$
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. $t := t+ 1$

Algoritm Flooding pentru consens

- Inițial: $x_j(0) = v_j \in R$
- Iterație locală:

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

Mai pe larg: actualizarea lui $x_i(t+1)$ se face pe baza mediei aritmetice dintre starea $x_i(t)$ și stările vecinilor $x_j(t), j \in \mathcal{N}_i^-$; presupunem un transfer cu succes al stărilor $x_j(t)$ către P_i. Vectorial avem:

$$\begin{bmatrix} x_1(t+1) \\ \dots \\ x_n(t+1) \end{bmatrix} = \begin{bmatrix} \frac{1}{d_1+1}\left(x_1(t) + \sum_{j \in \mathcal{N}_1^-} x_j(t)\right) \\ \dots \\ \frac{1}{d_n+1}\left(x_n(t) + \sum_{j \in \mathcal{N}_n^-} x_j(t)\right) \end{bmatrix} = \begin{bmatrix} \frac{1}{d_1+1}x_1(t) + \frac{1}{d_1+1}\sum_{j \in \mathcal{N}_1^-} x_j(t) \\ \dots \\ \frac{1}{d_n+1}x_n(t) + \frac{1}{d_n+1}\sum_{j \in \mathcal{N}_n^-} x_j(t) \end{bmatrix}$$

Observăm pe fiecare componentă a vectorului din partea dreaptă un produs scalar între stările nodurilor $\{i \cup \mathcal{N}_i^-\}$ și vectorul unidimensional $\tilde{a}_i = \frac{1}{d_i+1}[1 \ 1 \cdots 1]^T$. Sau, echivalent, între vectorul coloană definit de

$[a_i]_k = \begin{cases} \frac{1}{d_i+1}, k \in \{i \cup \mathcal{N}_i^-\} \\ 0, k \notin \{i \cup \mathcal{N}_i^-\} \end{cases}$ și vectorul stărilor $x(t)$.

Algoritm Flooding pentru consens

Algoritm Flooding de medie prezentat anterior se exprimă recurent prin actualizarea liniară:

$$x_i(t+1) = a_{ii}x_i(t) + \sum_{j \in \mathcal{N}_i^-} a_{ij}x_j(t) \qquad \forall i$$

Deci $x_i(t+1) = a_i^T x(t)$, iar dinamica stărilor sistemului este:

$$x(t+1) = Ax(t),$$

unde

$$A = \begin{bmatrix} a_1^T \\ \dots \\ a_n^T \end{bmatrix} \in R^{n \times n}, x(t) = \begin{bmatrix} x_1(t) \\ \dots \\ x_n(t) \end{bmatrix} \in R^n.$$

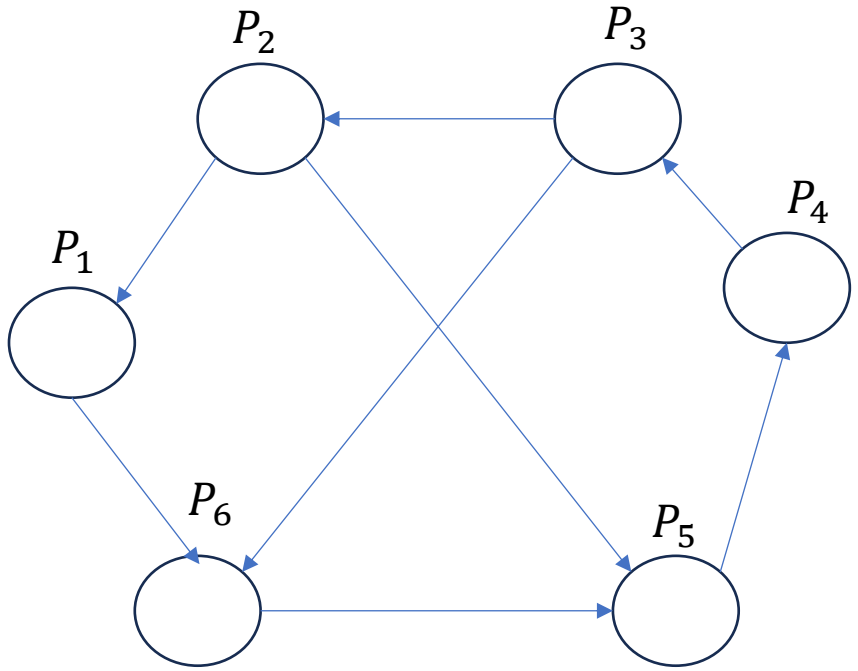
Matricea A este în strânsă legatură cu matricea de adiacență a grafului care determină topologia.

Care este structura matricii A pentru ponderi uniforme?

Exemple

Pentru graful alăturat matricea asociată este:

$$A = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 1/3 & 0 & 0 & 1/3 & 1/3 \\ 1/3 & 0 & 1/3 & 0 & 0 & 1/3 \end{bmatrix}$$



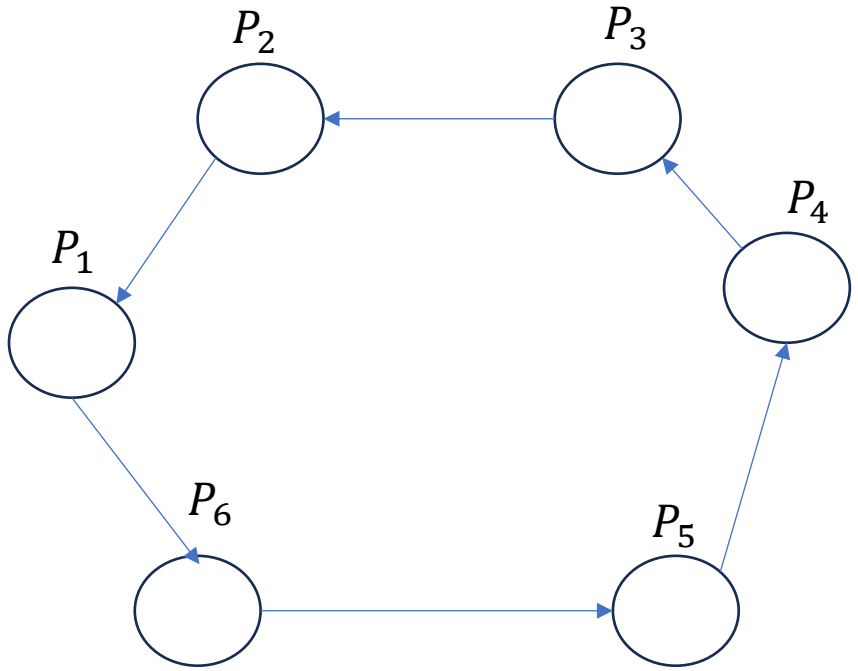
Definiție. Matricea A se numește *stohastică pe linii* dacă elementele $a_{ij} \geq 0$ și suma fiecărei linii este 1.

- Similar definim matricea stohastică pe coloane
- Matricile stohastice pe linii și coloane se numesc *dublu stochastice*.
- Exemplul de mai sus unde se încadrează?
- Alte exemple? Cum generăm matrici dublu stohastice?

Exemple

Pentru un inel de dimensiune $n = 6$, matricea asociată este:

$$A = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 1/2 & 0 & 0 & 0 & 0 & 1/2 \end{bmatrix}$$

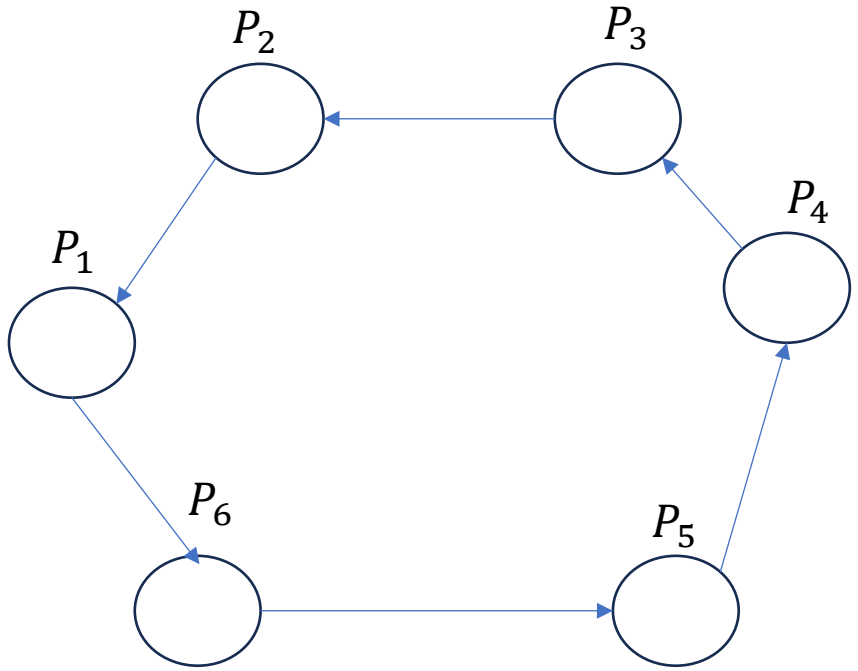


Exemple

În general, pentru un inel de dimensiune n, matricea asociată este:

$$A = \begin{bmatrix} 1/2 & 1/2 & 0 & \cdots & 0 & 0 \\ 0 & 1/2 & 1/2 & & 0 & 0 \\ & \vdots & & \ddots & \vdots & \\ 1/2 & 0 & 0 & \cdots & 0 & 1/2 \end{bmatrix} \in R^{n \times n}$$

- Dublu stohastică
- Nesimetrică (graf directat)

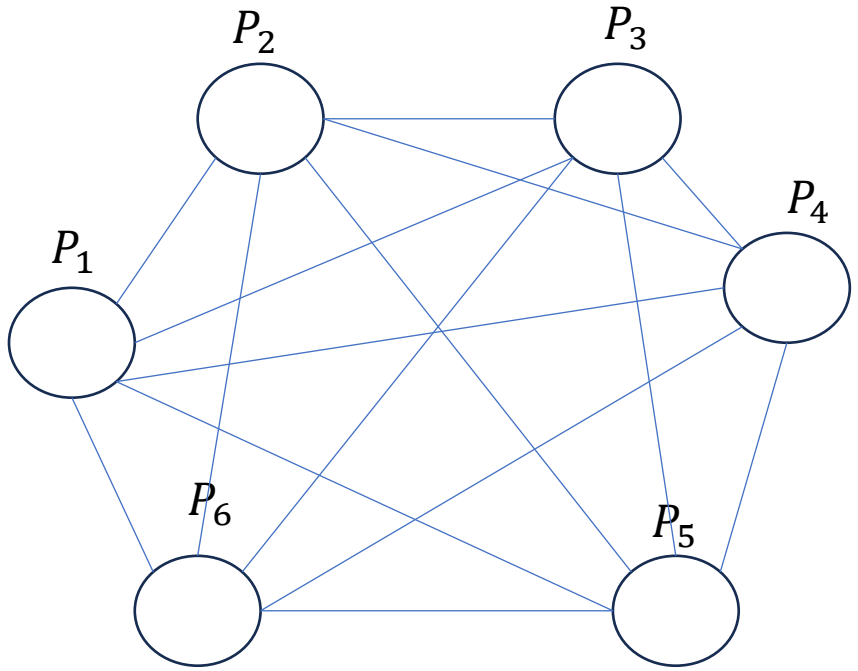


Exemple

Pentru un graf de tip clică matricea asociată este:

$$A = \begin{bmatrix} 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \end{bmatrix}$$

- Dublu stohastică
- Simetrică



Algoritm Flooding pentru consens

Din dinamică stărilor

$$x(t+1) = Ax(t),$$

se observă ușor:

$$x(t) = A^t x(0),$$

de aceea convergența depinde total de comportamentul matricii A^t (implicit, doar de structura grafului).

Teorema. Dacă matricea A este stohastică pe linii, atunci se atinge consensul asimptotic, i.e. $x(t) \to c\mathbf{1}$ când $t \to \infty$. În plus, dacă matricea A este stohastică pe coloane (graful are grade de intrare uniforme), i.e. $\mathbf{1}^{\mathbf{T}} A = A^T$, atunci

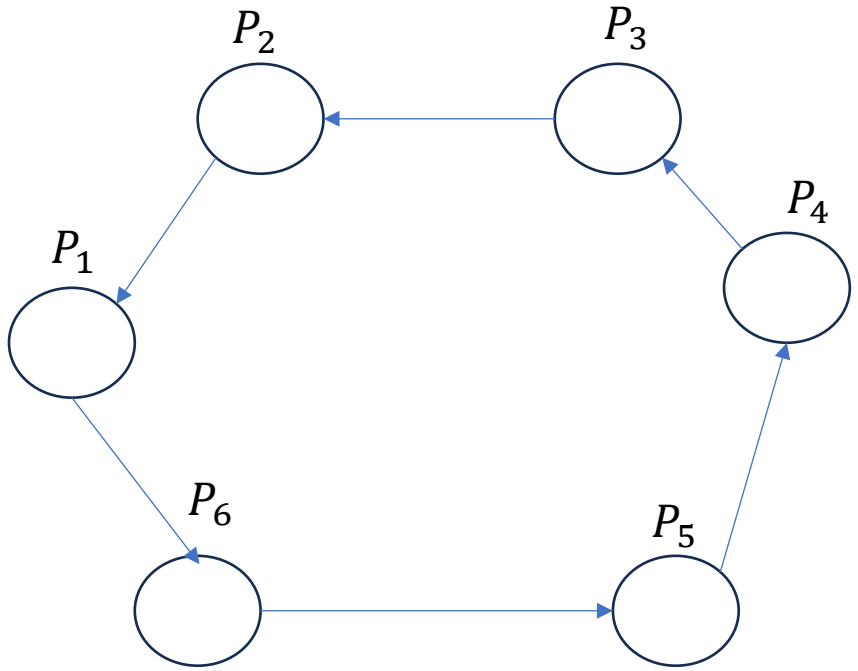
$$c = \frac{1}{n}\sum_{i=1}^{n} x_i(0)\,.$$

- Rezultat valabil nu doar pentru ponderi uniforme.
- Condiția necesară pentru consens este ca matricea ponderilor să fie stohastică pe linii (fiecare să realizeze la fiecare iterație o combinație convexă între starea proprie și stările vecinilor)
- Dacă matricea ponderilor este, în plus, stohastică pe coloane, atunci valoarea de consens este media aritmetică a stărilor inițiale.

Exemple

Pentru un inel de dimensiune $n = 6$, matricea asociată este:

$$A = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 1/2 & 0 & 0 & 0 & 0 & 1/2 \end{bmatrix}$$

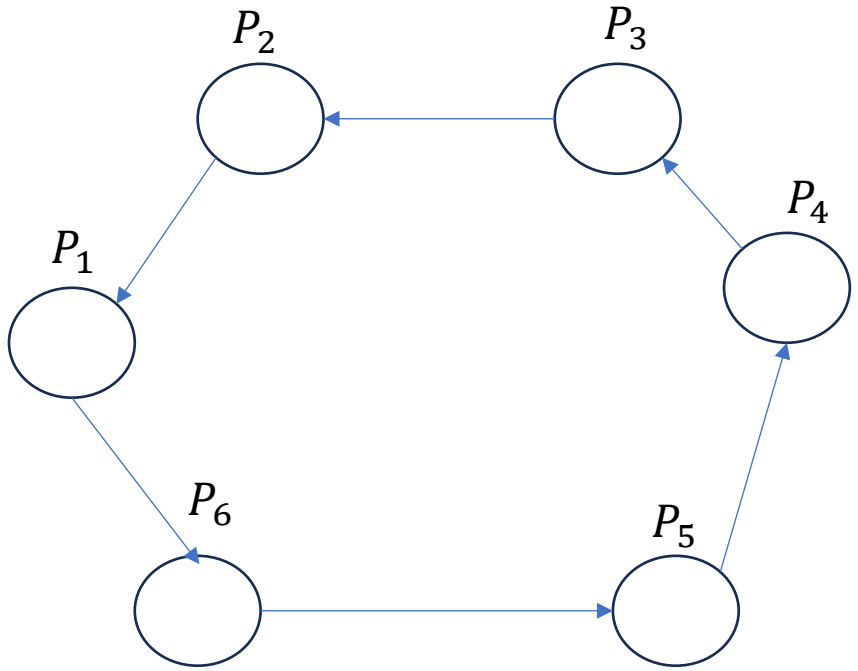



```
A^2:
   0.2500   0.5000   0.2500         0         0         0
        0   0.2500   0.5000   0.2500         0         0
        0         0   0.2500   0.5000   0.2500         0
        0         0         0   0.2500   0.5000   0.2500
   0.2500         0         0         0   0.2500   0.5000
   0.5000   0.2500         0         0         0   0.2500
```

Exemple

Pentru un inel de dimensiune $n = 6$, matricea asociată este:

$$A = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 1/2 & 0 & 0 & 0 & 0 & 1/2 \end{bmatrix}$$

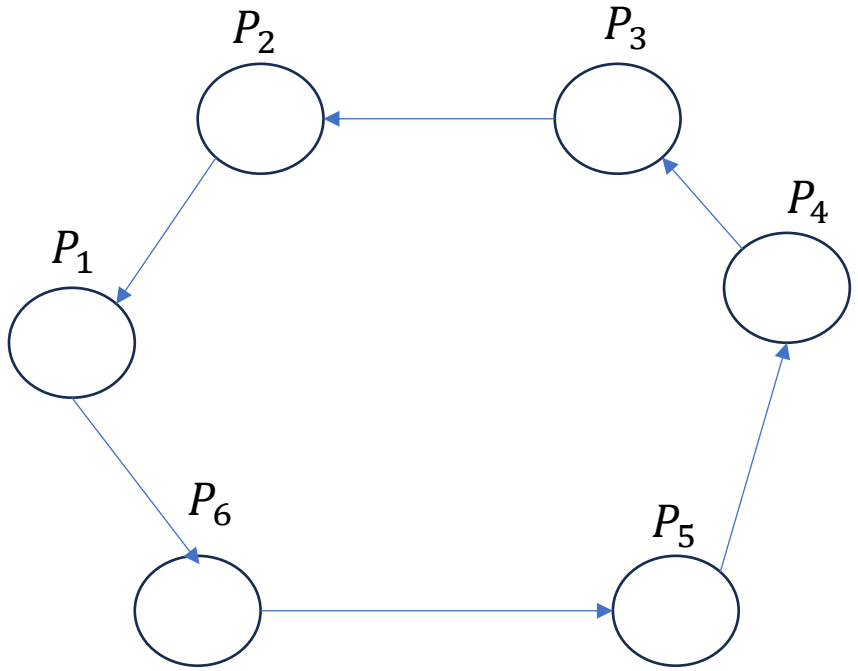



```
A^4:
   0.0625   0.2500   0.3750   0.2500   0.0625        0
        0   0.0625   0.2500   0.3750   0.2500   0.0625
   0.0625        0   0.0625   0.2500   0.3750   0.2500
   0.2500   0.0625        0   0.0625   0.2500   0.3750
   0.3750   0.2500   0.0625        0   0.0625   0.2500
   0.2500   0.3750   0.2500   0.0625        0   0.0625
```

Exemple

Pentru un inel de dimensiune $n = 6$, matricea asociată este:

$$A = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 1/2 & 0 & 0 & 0 & 0 & 1/2 \end{bmatrix}$$

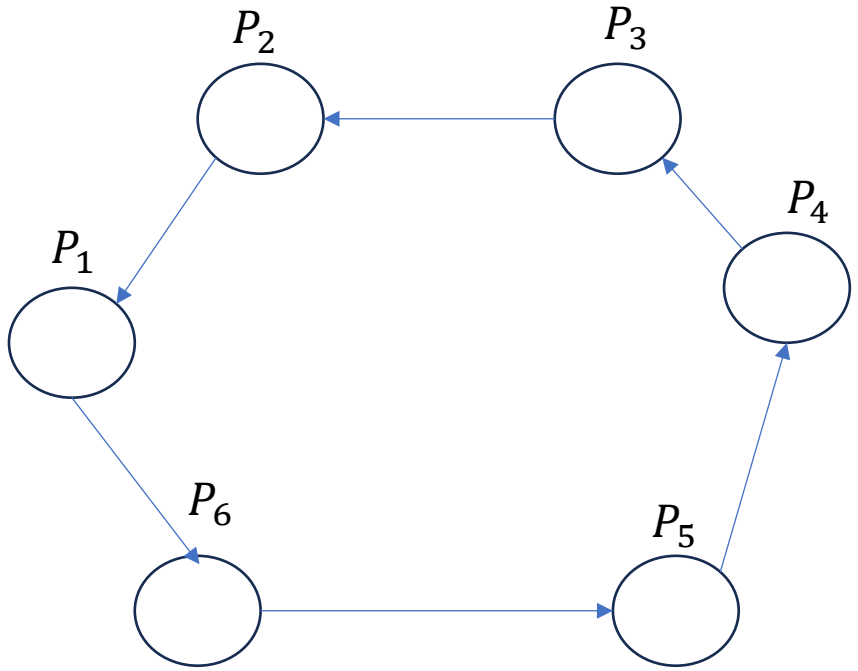



```
A^10:
   0.206055   0.126953   0.087891   0.126953   0.206055   0.246094
   0.246094   0.206055   0.126953   0.087891   0.126953   0.206055
   0.206055   0.246094   0.206055   0.126953   0.087891   0.126953
   0.126953   0.206055   0.246094   0.206055   0.126953   0.087891
   0.087891   0.126953   0.206055   0.246094   0.206055   0.126953
   0.126953   0.087891   0.126953   0.206055   0.246094   0.206055
```

Exemple

Pentru un inel de dimensiune $n = 6$, matricea asociată este:

$$A = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 1/2 & 0 & 0 & 0 & 0 & 1/2 \end{bmatrix}$$

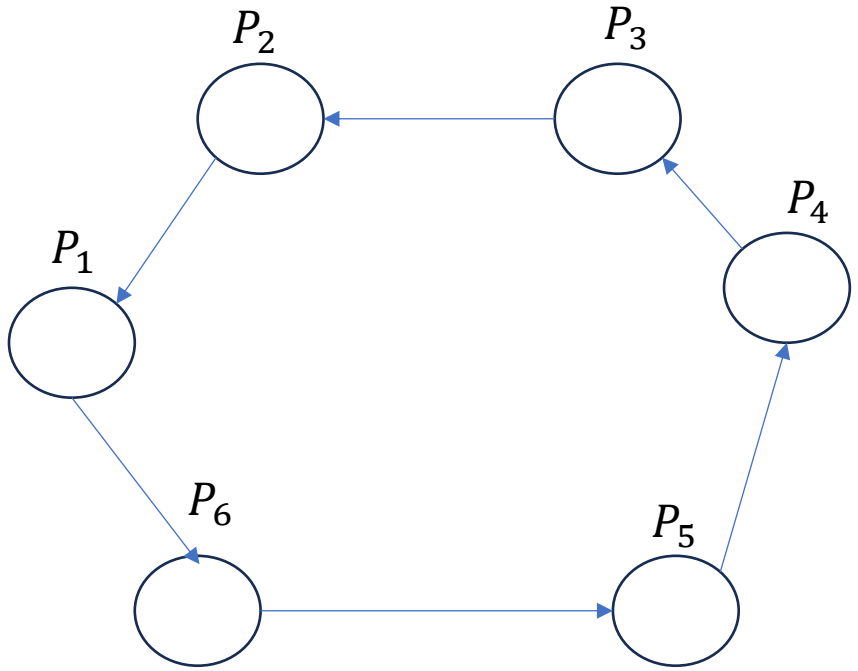



```
A^25:
   0.1746   0.1746   0.1667   0.1587   0.1587   0.1667
   0.1667   0.1746   0.1746   0.1667   0.1587   0.1587
   0.1587   0.1667   0.1746   0.1746   0.1667   0.1587
   0.1587   0.1587   0.1667   0.1746   0.1746   0.1667
   0.1667   0.1587   0.1587   0.1667   0.1746   0.1746
   0.1746   0.1667   0.1587   0.1587   0.1667   0.1746
```

Exemple

Pentru un inel de dimensiune $n = 6$, matricea asociată este:

$$A = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 1/2 & 0 & 0 & 0 & 0 & 1/2 \end{bmatrix}$$

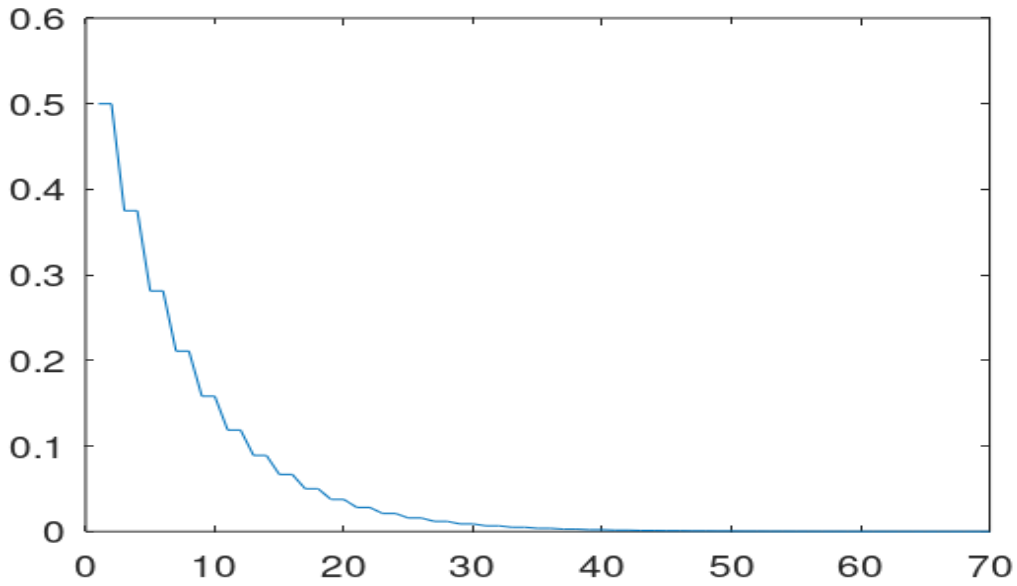


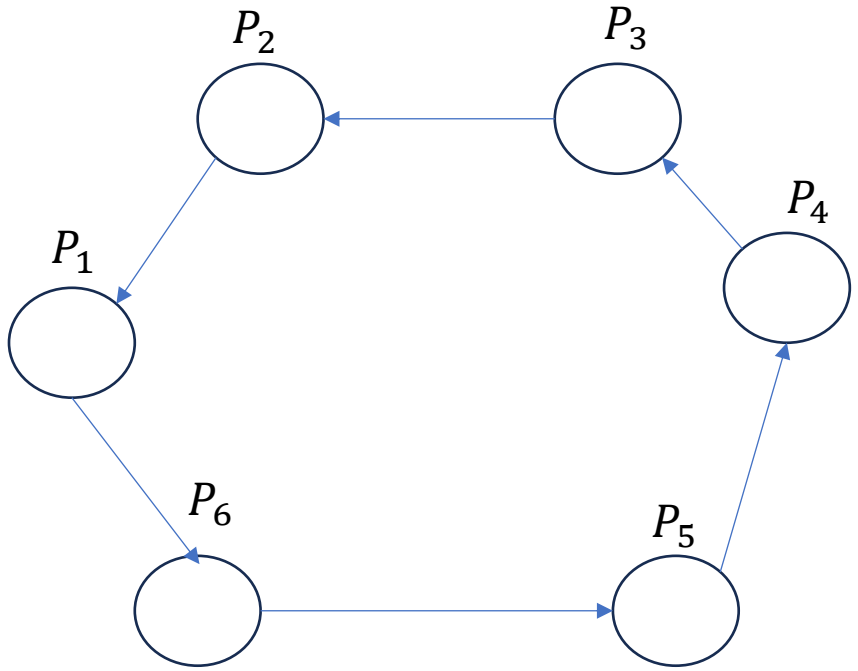

```
A^70:
   0.1667   0.1667   0.1667   0.1667   0.1667   0.1667
   0.1667   0.1667   0.1667   0.1667   0.1667   0.1667
   0.1667   0.1667   0.1667   0.1667   0.1667   0.1667
   0.1667   0.1667   0.1667   0.1667   0.1667   0.1667
   0.1667   0.1667   0.1667   0.1667   0.1667   0.1667
   0.1667   0.1667   0.1667   0.1667   0.1667   0.1667
```

Exemple

Pentru un inel de dimensiune $n = 6$, matricea asociată este:

$$A = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 1/2 & 0 & 0 & 0 & 0 & 1/2 \end{bmatrix}$$





$\mathrm{V}(t) = \max\big(x(t)\big) - \min(x(t))$

Plot: $\mathrm{V}(t)$ vs. t

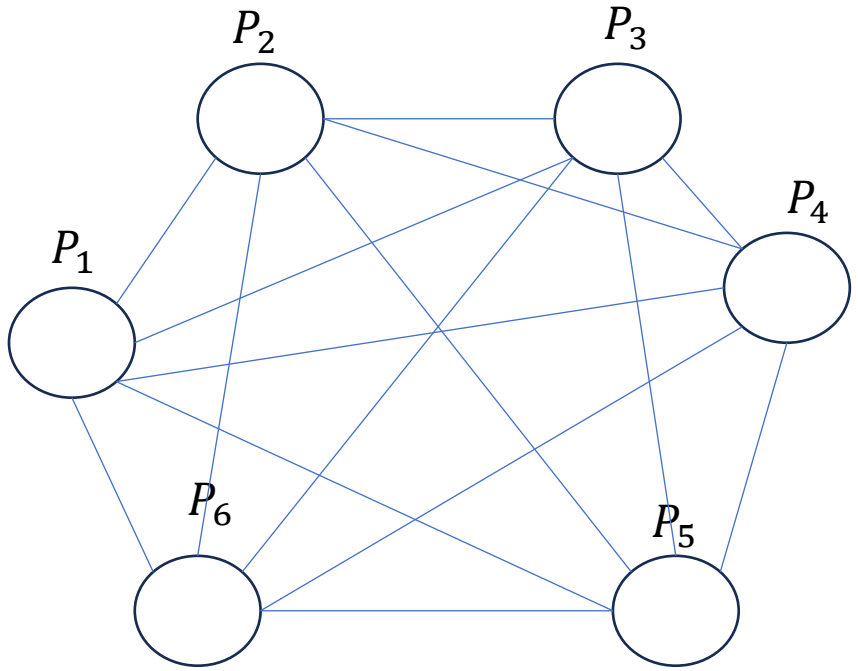
Exemple

Pentru un graf de tip clică matricea asociată este:

$$A = \begin{bmatrix} 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \end{bmatrix}$$

$$x(1) = x^*$$

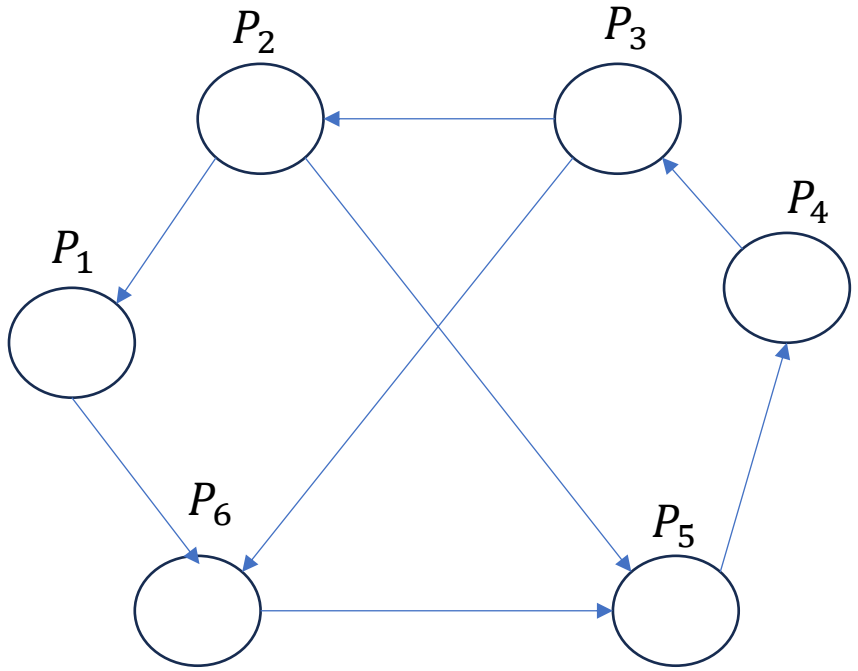
- Convergență într-un singur pas



Exemple

Pentru graful alăturat matricea asociată este:

$$A = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 1/3 & 0 & 0 & 1/3 & 1/3 \\ 1/3 & 0 & 1/3 & 0 & 0 & 1/3 \end{bmatrix}$$

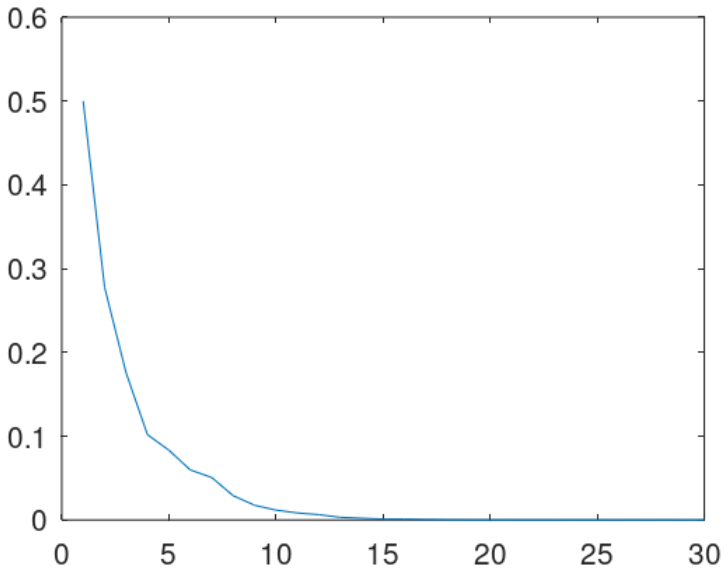


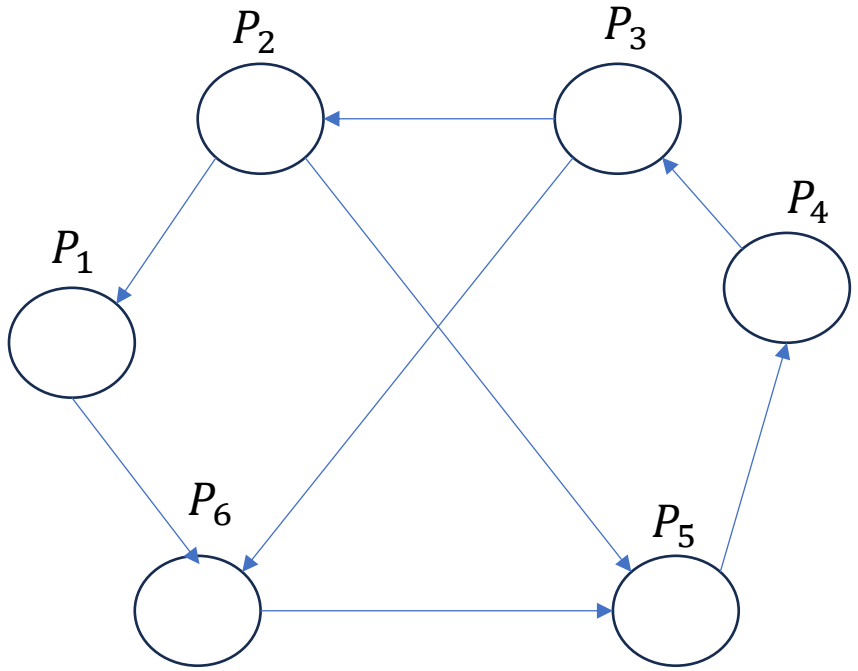

```
A^30:
   0.060607   0.181818   0.242425   0.242424   0.181817   0.090909
   0.060606   0.181818   0.242424   0.242425   0.181818   0.090909
   0.060606   0.181818   0.242424   0.242424   0.181819   0.090909
   0.060606   0.181818   0.242424   0.242424   0.181818   0.090909
   0.060606   0.181818   0.242424   0.242424   0.181818   0.090909
   0.060606   0.181818   0.242424   0.242424   0.181818   0.090909
```

Exemple

Pentru graful alăturat matricea asociată este:

$$A = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 1/3 & 0 & 0 & 1/3 & 1/3 \\ 1/3 & 0 & 1/3 & 0 & 0 & 1/3 \end{bmatrix}$$





$\mathrm{V}(t) = \max\big(x(t)\big) - \min(x(t))$

Plot: $\mathrm{V}(t)$ vs. t

Defecte

Defecte

- Un număr sporit de noduri implică o probabilitate de defecte în creștere.
- Gravitatea defectului depinde de aplicație: sistem de control al traficului aerian vs sistem de gaming.
- Sursele defectelor la nivel de nod:
  - Erori: design, fabricație, programare
  - Accidente fizice
  - Condiții de mediu dure
  - Date de intrare neașteptate
  - Etc.

Modelarea defectelor

Pentru a evita restartarea algoritmilor de fiecare dată când apare un defect (sau adăugarea permanentă de resurse), sunt necesare *schemele (algoritmii) tolerante la defecte*.

Principalele modele:

- Defect *Crash*
- Defect *Bizantin*

Nodurile care nu suferă defect se numesc noduri **corecte.**

Redundanță

- Redundanța informației:
  - Adaugă extra biți pentru recuperarea datelor; tehnici ECC, coduri Hamming; checksum / CRC – pentru detecția alterării mesajelor.
- Redundanța de timp:
  - O acțiune executată va fi repetată dacă este necesar (E.g. Re-transmiterea pachetelor, tranzacții atomice, etc)
- Redundanța fizică:
  - Echipament extra adăugat în sistem pentru a tolera eventualele defecte.
  - Două moduri de organizare:
    a) Active replication
    b) Primary backup

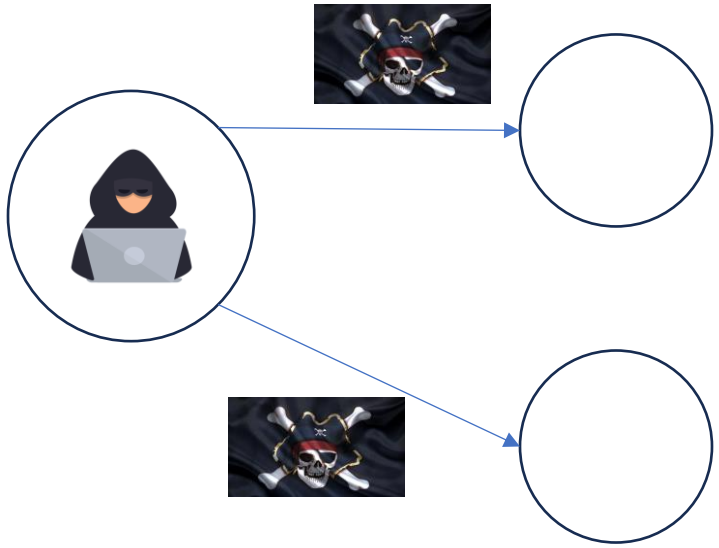
Defectul Crash

- La momentul defectului, nodul:
  - Oprește execuția locală a iterațiilor
  - Nu primește mesaje
  - Nu trimite mesaje
- În perspectiva cea mai simplistă: nodul nu reia activitatea niciodată.
- Sub-clase de Crash:
  - Crash-stop
  - Omisiune de mesaje
  - Crash cu revenire

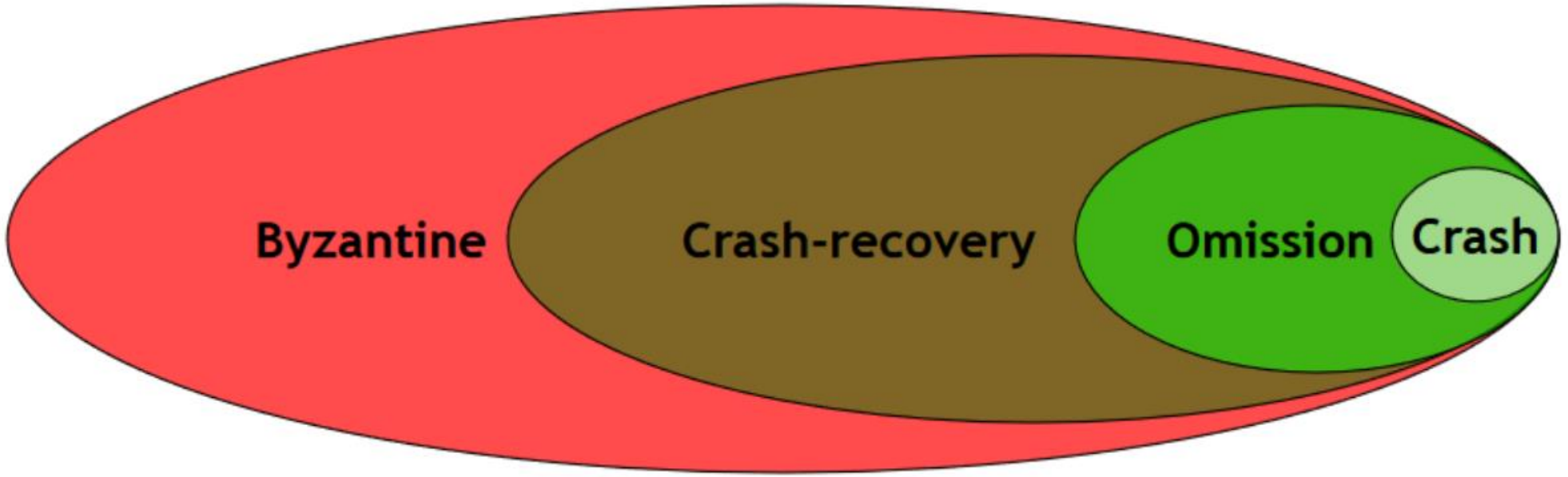
Defect Bizantin

Sub defect bizantin nodurile se comportă malițios, perturbând activitatea întregului sistem (e.g. comportament arbitrar):

- Livrează mesaje atipice execuției algoritmului local
- Actualizează starea după reguli atipice execuției algoritmului local



Ierarhie



Seif Haridi, Distributed Systems course, https://canvas.instructure.com/courses/902299/files/31539484?module_item_id=6616396

Consens distribuit (cu procese defecte)

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 (algoritm): Într-un algoritm de consens, orice nod *corect* din sistem va decide eventual la un moment de timp.

În general, decizia se reduce la evaluarea funcției de consens f(·) în $x(0)$.

Consens distribuit (cu procese defecte)

Ipoteze:

- Graf complet (nedirectat)
- Un nod poate fi corect sau defect *Crash* (încetează funcționarea)
- Dacă un nod intră în starea de defect, rămâne în ea până la finalul algoritmului

Urmărim atingerea consensului distribuit sub maxim $s \geq 0$ defecte *Crash*.

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. **Bcast**($< x_i$ (0), $id_i >$)
2. Fie U mulțimea mesajelor $< v_j, id_j >$ primite restul nodurilor
3. $M_i(t+1) = M_i(\mathrm{t}) \cup U$
4. Fie $V_i(\mathrm{t}+1)$ mulțimea valorilor v_j din $M_i(t+1)$
5. Fie $I_i(\mathrm{t}+1)$ mulțimea id-urilor id_j din $M_i(t+1)$
6. **Return** $f(V_i(\mathrm{t}))$

Algoritm FloodSet pentru clică

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. **Bcast**($< x_i\ (0), id_i >$)
2. Fie U mulțimea mesajelor $< v_j, id_j >$ primite restul nodurilor
3. $M_i(t+1) = M_i(\mathrm{t}) \cup U$
4. Fie $V_i(\mathrm{t}+1)$ mulțimea valorilor v_j din $M_i(t+1)$
5. Fie $I_i(\mathrm{t}+1)$ mulțimea id-urilor id_j din $M_i(t+1)$
6. **Return** $f(V_i(\mathrm{t}))$

$j \in \mathcal{N}_i^-$ cade la momentul t, atunci $M_j(t)$ nu va ajunge la P_i la momentul $t+1$

$V_i \neq V_k$

Algoritm FloodSet s −robust (clică, defect Crash)

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

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. **Bcast**(M_i (t))
2. Fie U mulțimea mesajelor $< v_j, id_j >$ primite restul nodurilor
3. $M_i(t+1) = M_i(\text{t}) \cup U$
4. Fie V_i(t+1) mulțimea valorilor v_j din $M_i(t+1)$
5. Fie I_i(t+1) mulțimea id-urilor id_j din $M_i(t+1)$
6. **If** $(t > s+1)$:
    1. **Return** $f(M_i(\text{t}))$
7. $t := t+1$

- Paradigma de ‘’robustificare’’ a algoritmilor distribuiti
- Se realizeaza s + 1 iteratii (s explicit)
- **Lemma 1**. Dacă există o iterație t în care nu există defect, atunci $M_i(t) = M_j(t)$ pentru orice noduri i și j corecte la momentul t.
- **Lemma 2**. Dacă $M_i(t) = M_j(t)$ pentru orice noduri i și j corecte. Atunci pentru orice $t \leq t' \leq f+1$ avem $M_i(t') = M_j(t')$ pentru orice noduri i și j corecte la momentul t'.
- **Teorema**. FloodSet s −robust rezolvă problema de consens pentru defecte de tip *Crash*.
- Principalul argument: avem s defecte, de aceea dupa $s+1$ iterații va exista cel puțin o iterație t în care nu există defect. Lemma 1 implică $M_i(t) = M_j(t)$ pentru orice noduri i și j corecte la momentul t. Lemma 2 implică $M_i(s+1) = M_j(s+1)$ pentru orice noduri i și j corecte la momentul $s+1$.

Algoritm FloodSet s −robust (clică, defect Crash)

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

M_i: - int id (id propriu)
- int v (token, inițial egal cu x_i)
- funcție obiectiv f ()
- int t, integer, inițial 0

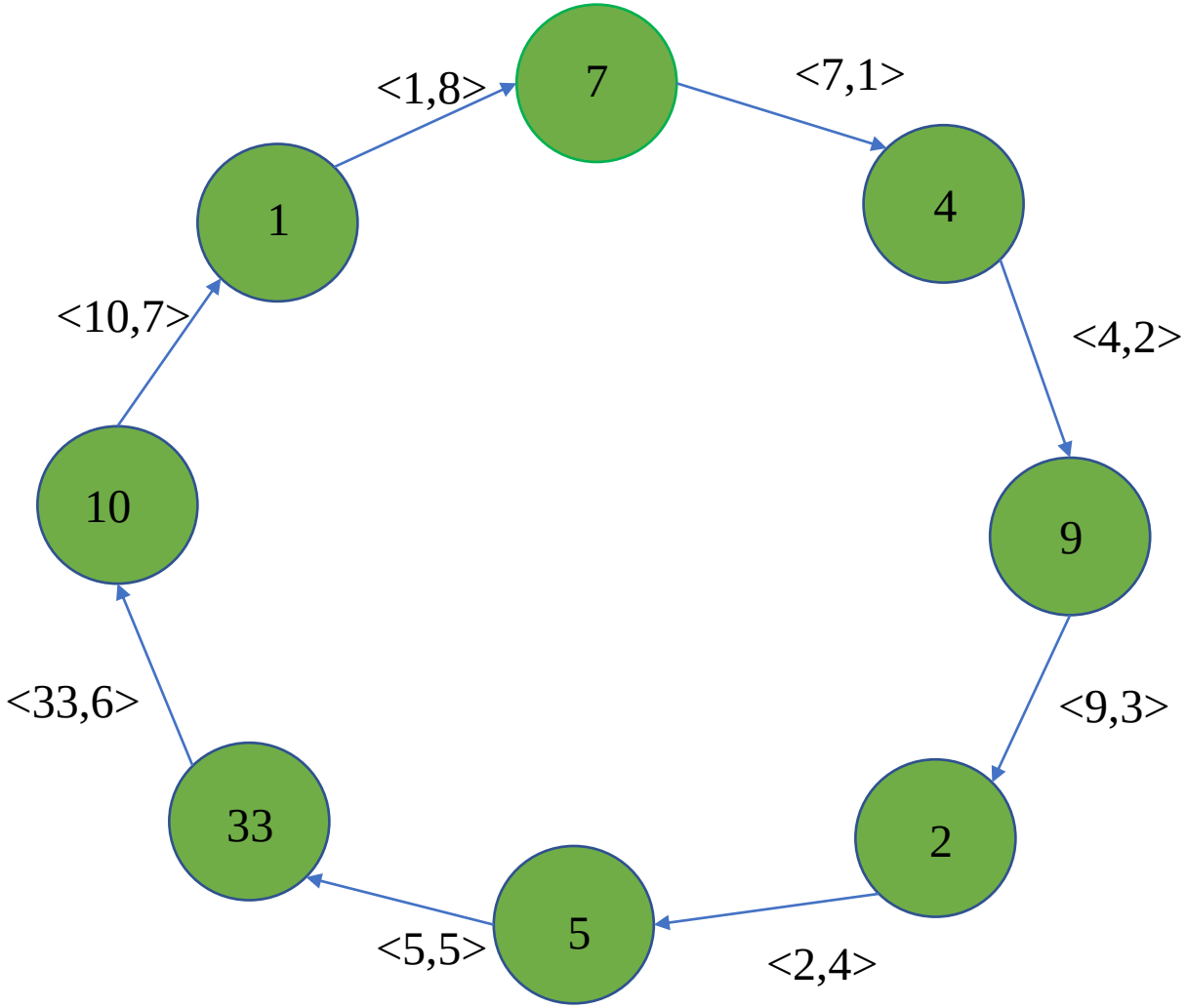
Funcție transformare nod i ():

1. **Bcast**(M_i (t))
2. Fie U mulțimea mesajelor $< v_j, id_j >$ primite restul nodurilor
3. $M_i(t+1) = M_i(\text{t}) \cup U$
4. Fie V_i(t+1) mulțimea valorilor v_j din $M_i(t+1)$
5. Fie I_i(t+1) mulțimea id-urilor id_j din $M_i(t+1)$
6. **If** $(t > s+1)$:
    1. **Return** $f(M_i(\text{t}))$
7. $t := t+1$

- Operatia de Broadcast costa n mesaje
- Complexitate timp: $\text{s} + 1$
- Complexitate mesaj: $(\text{s} + 1)n$
- Rezultat margine inferioara (informal)[Lynch]: orice algoritm s-robust la defecte Crash executa minim s+1 iteratii
- Cum generalizăm pentru grafuri conexe generale (nu neaparat complete)?

Algoritm FloodSet s −robust

Iterația 1

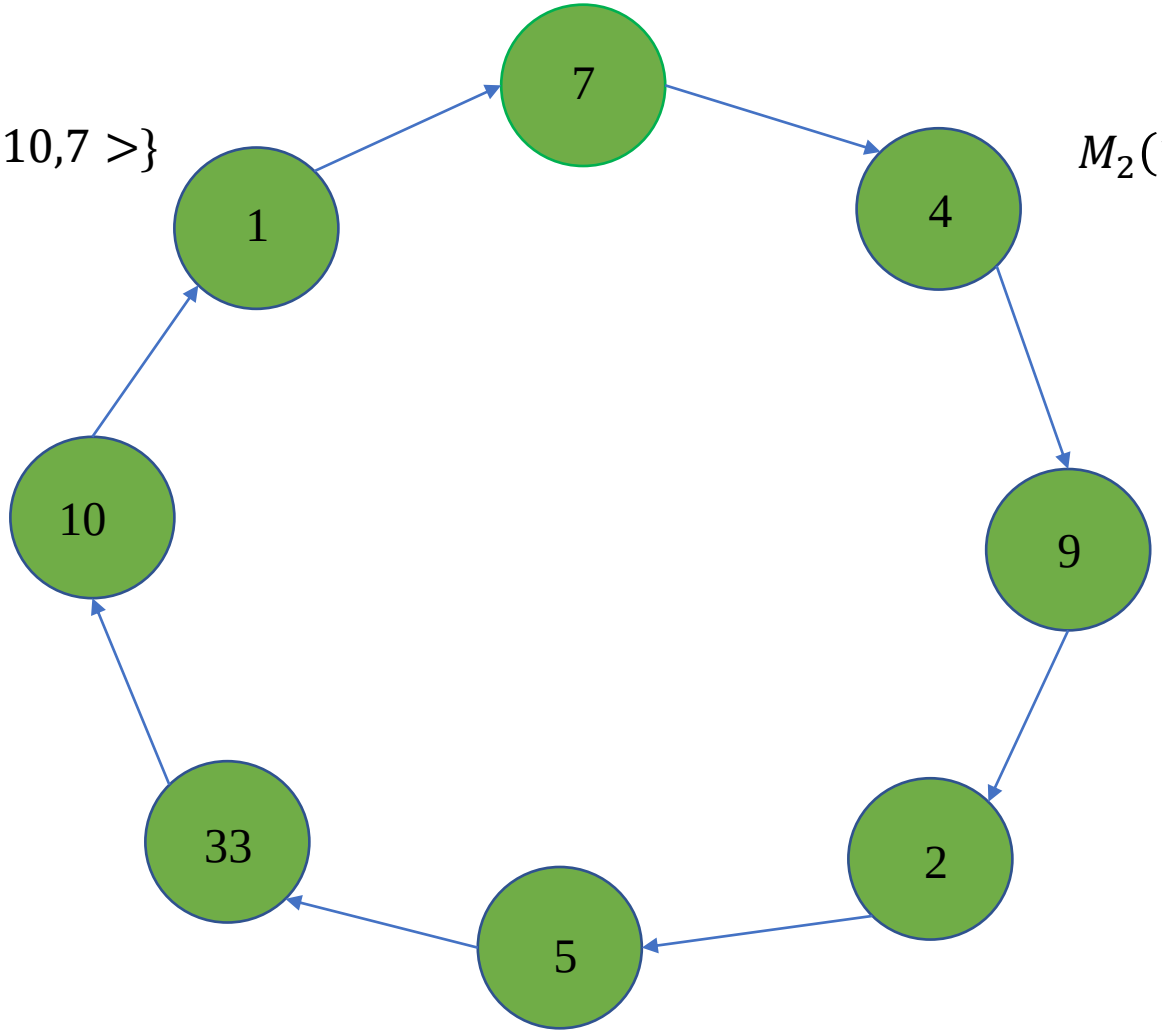


Algoritm FloodSet s −robust

Iterația 1

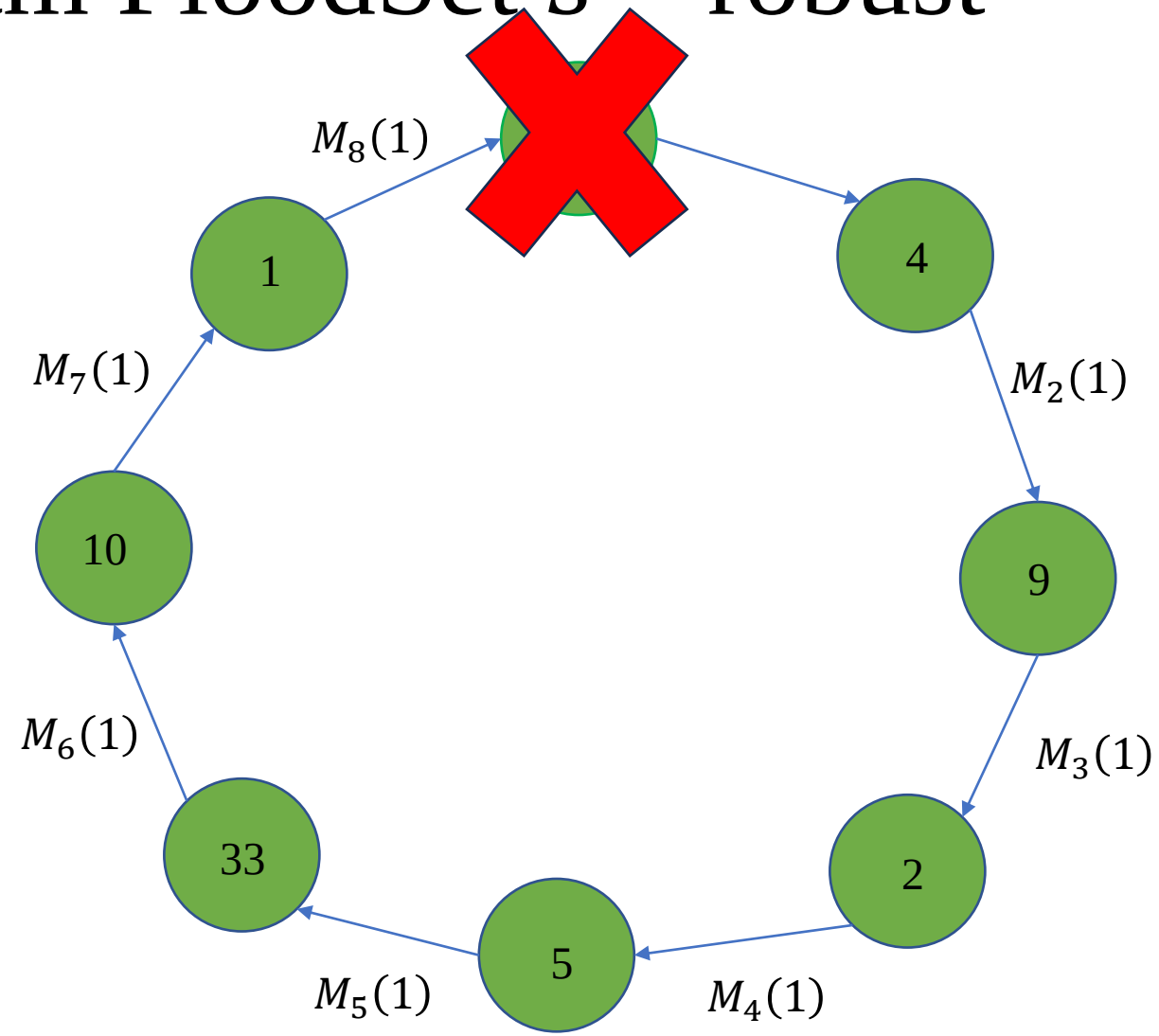
$M_8(1) = \{< 1,8 >, < 10,7 >\}$

$M_2(1) = \{< 4,2 >, < 7,1 >\}$



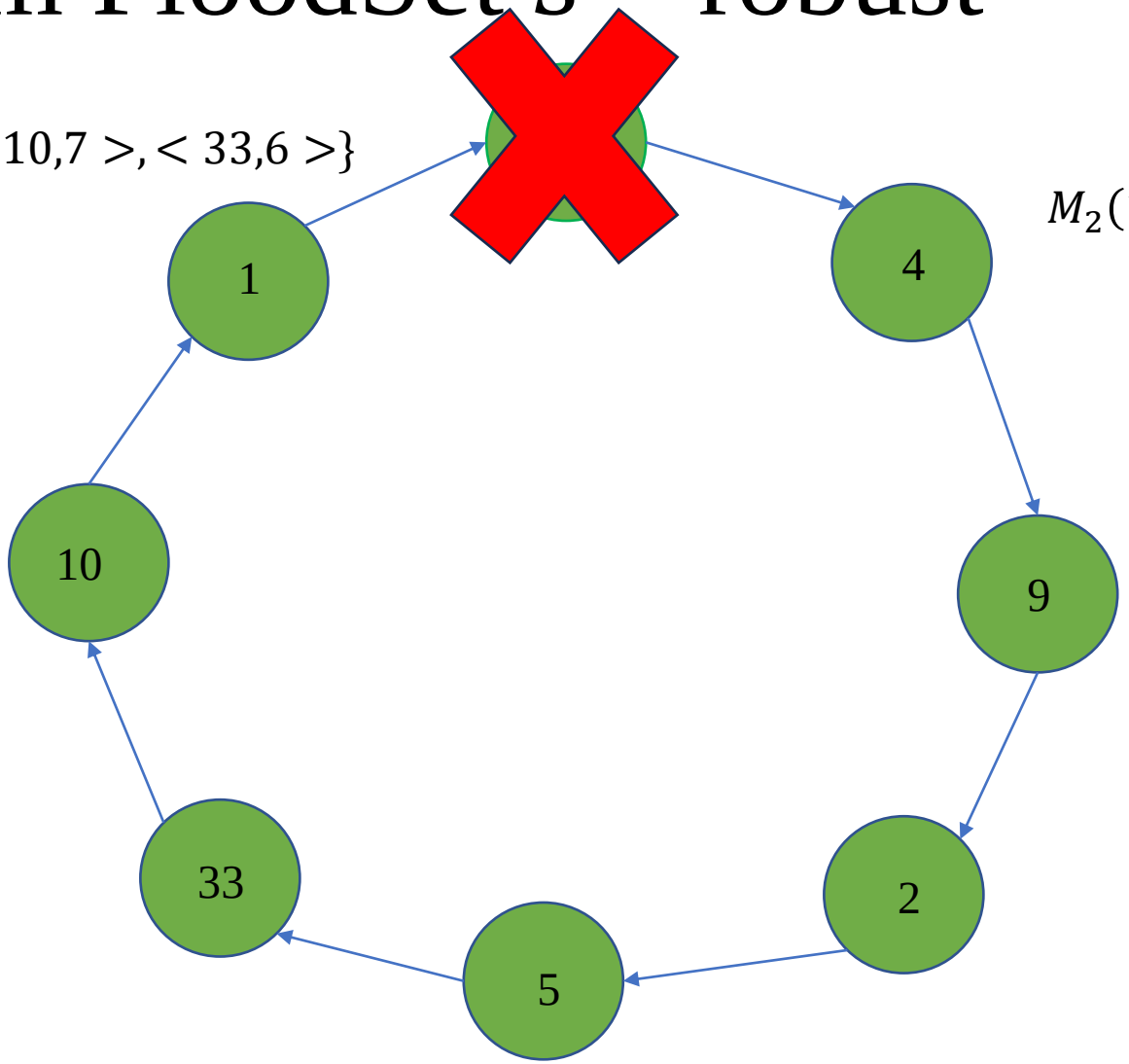
Algoritm FloodSet s −robust

Iterația 2



Algoritm FloodSet s −robust

$M_8(1) = \{< 1,8 >, < 10,7 >, < 33,6 >\}$



Iterația 2

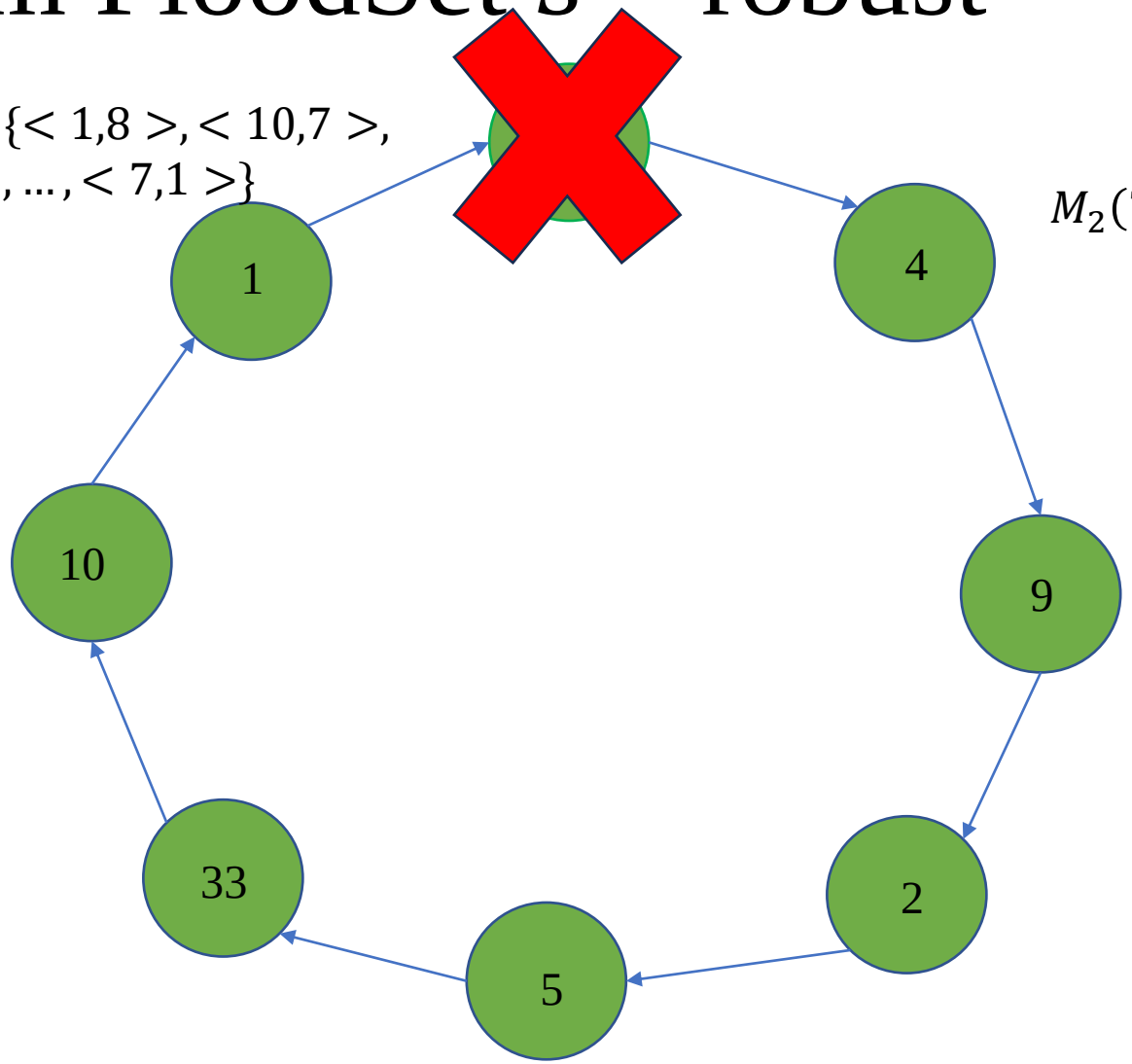
$M_2(1) = \{< 4,2 >, < 7,1 >\}$

Algoritm FloodSet s −robust

$M_8(7) = \{< 1,8 >, < 10,7 >, < 33,6 >, \dots, < 7,1 >\}$

Iterația 7

$M_2(7) = \{< 4,2 >, < 7,1 >\}$



Consens distribuit (cu procese defecte)

Ipoteze:

- Graf ~~complet~~ conex (nedirectat). Conectivitatea grafului $G, conn(G)$ = numărul minim de noduri care, o dată eliminate din graf, va rezulta un graf neconectat.
- $s < conn(G)$
- Un nod poate fi corect sau defect *Crash* (încetează funcționarea)
- Dacă un nod intră în starea de defect, rămâne în ea până la finalul algoritmului

Urmărim atingerea consensului distribuit sub maxim $s \geq 0$ defecte *Crash*.

Algoritm FloodSet s −robust (conex, defect Crash)

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 $< v_j, id_j >$ primite de la $\mathcal{N}_i^-$
2. $M_i(t+1) = M_i(\text{t}) \cup U$
3. Fie V_i(t+1) mulțimea valorilor v_j din $M_i(t+1)$
4. Fie I_i(t+1) mulțimea id-urilor id_j din $M_i(t+1)$
5. **If** $(t > (s+1)diam)$:
   1. **Return** $f(M_i(\text{t}))$
6. **Else**: send(M_i(t+1), $\mathcal{N}_i^+$)
7. $t := t+1$

- Ipoteza $s < conn(G)$ garantează că graful rezultat în urma defectelor rămâne conex.
- Se realizează $s+1$ seturi de $diam(G)$ iterații.
- Convergența folosește aceleași argumente ca în cazul clicii; în principal, dupa $s+1$ seturi de $diam(G)$ iterații, există cel puțin unul în care niciun nod nu are defect. Însa $diam(G)$ sunt suficiente pentru a atinge consensul între nodurile corecte.