# Cloud Computing

## Cap 3. Membership protocol

October 29, 2022

1. Detectorul de failures optimal
2. Protocolul scalabil de tip infection-style
3. Mecanismul suspicion

# Detectorul de failures optimal (1)

Are următoarele caracteristici măsurabile:

- completness
  - detectarea tuturor căderilor
- accuracy
  - proprietatea de a nu genera alarme false (noduri raportate ca picate atunci când nu este cazul)
- viteza
  - timpul până la prima detecție a căderii nodului
- scalabilitate
  - încărcare echilibrată a nodurilor
  - încărcarea rețelei

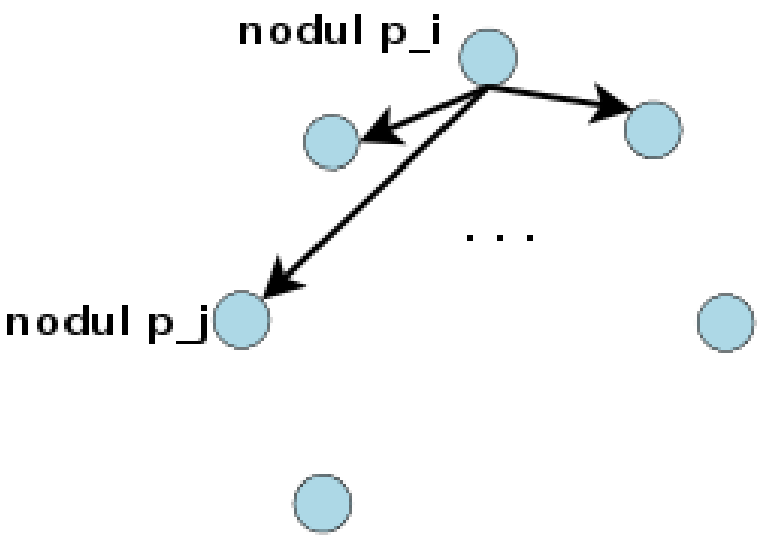
# Detectorul de failures optimal (2)

- completness **: garantat întotdeauna**
- accuracy : **probabilitatea de a detecta greșit o cădere în T unități de timp,** $PM(T)$
- viteza **: T unități de timp**
  - timpul până la prima detecție a căderii nodului
- scalabilitate
  - încărcare echilibrată a nodurilor
  - încărcarea rețelei: **N noduri**

# Detectorul de failures optimal (3)

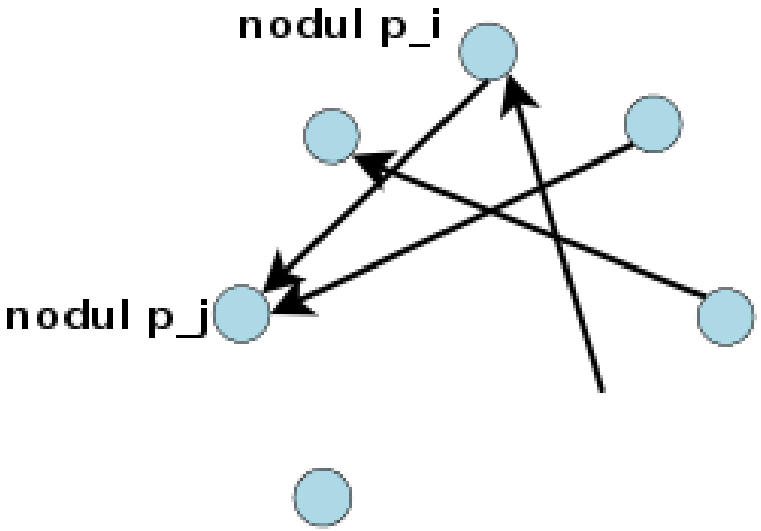
- completness : garantat întotdeauna
- accuracy : probabilitatea de a detecta greșit o cădere în T unități de timp, $PM(T)$
- viteza : T unități de timp
  - timpul până la prima detecție a căderii nodului
- scalabilitate
  - încărcare echilibrată a nodurilor
  - L - încărcarea unui nod
  - încărcarea rețelei **: N * L, comparație pentru toate protocoalele**

# Heartbeating de tip all-to-all




- se trimit heartbeats din T în T secunde
- N este numărul de noduri
- perioada de timeout e un multiplu de T
- încărcarea nodului este $L = N/T$, se trimit N intrări de tabelă pentru fiecare nod - O(N)
- un nod trimite mesajul la N noduri în timp O(1), altfel încărcarea nodului poate să fie și $O(N^2)$ (trimiterea listei de membership de lungime N la N noduri în perioada T)

# Heartbeating de tip gossip




- perioada este $T = log(N) * t_{gossip}$
- $t_{gossip}$ este perioada la care un nod trimite către un număr fix de noduri selectate
- $L = N/t_{gossip}$ (N dă lungimea listei de membership; întreaga listă de membership se trimite către un număr fix de noduri selectate)

  $L = N * log(N)/T$
- încărcare mai mare

# Cazul optim (1)

- T = timpul de propagare
- PM(T) = probabilitatea de detecție greșită (mistake) a căderii
- N = numărul de noduri
- $p_{ml}$ = probabilitatea message loss, independentă
- considerăm $m$ mesaje
- probabilitatea ca toate să se piardă este $p_{ml}^m$
- pt. ca măcar unul din cele $m$ mesaje trimise să ajungă,

$$PM(T) \leq p_{ml}^m \qquad m \geq \frac{log(PM(T))}{log(p_{ml})}$$

$$L^* \geq \frac{log(PM(T))}{log(p_{ml})} \cdot \frac{1}{T}$$

# Cazul optim (2)

- în cazul optim încărcarea per nod $L^*$ este independentă de N, O(1) !
- all-to-all și gossip: sub-optimale
  - L = cel puțin O(N / T)
  - detecția simultană a căderilor în toate nodurile
  - nu distinge între detecția căderilor și diseminare

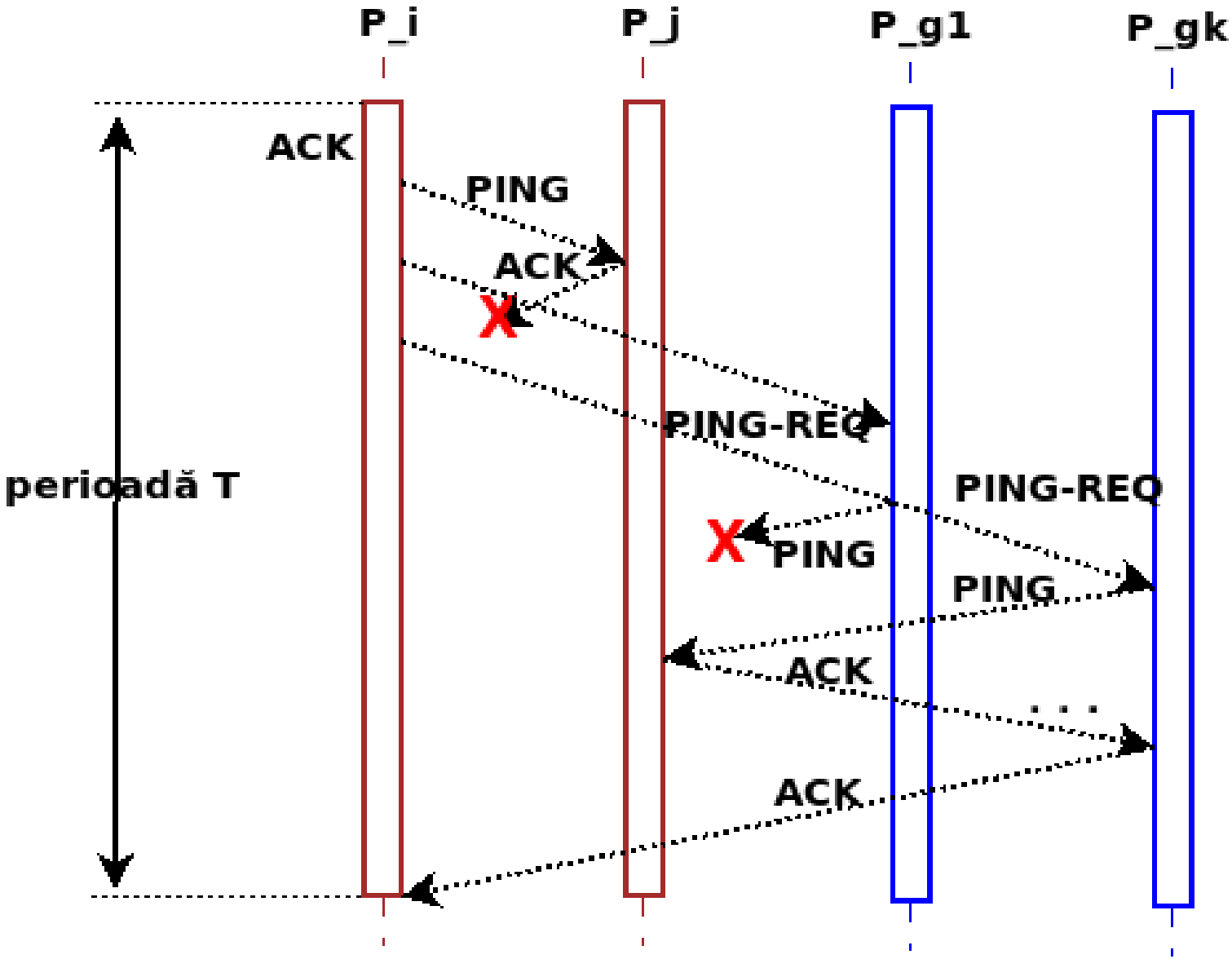
* e necesară separarea componentei de detecție de cea de diseminare

* folosirea unei componente de failure ce nu e bazată pe heartbeat

# SWIM failure detection (1)

- "SWIM: Scalable Weakly-consistent Infection-style Process Group Membersip Protocol", A. Das, I. Gupta, A. Motivala
  `https://www.cs.cornell.edu/~asdas/research/dsn02-swim.pdf`
- folosește mesaje ping și ACK
- procesele $p_j$, $p_{g_1}$, ..., $p_{g_k}$ sunt alese aleator de către procesul $p_i$

# SWIM failure detection (2)



# SWIM failure detection (3)

| protocol | detection time | process load |
|---|---|---|
| gossip heartbeating cu bandă limitată | O(N) | constant |
| gossip heartbeating cu bandă în O(N) | constant | O(N) |
| SWIM | constant | constant |

# SWIM failure detection (4)

| Parametru | Valoare |
|---|---|
| Timpul primei detecții | valoarea așteptată = $\frac{e}{e-1}$ perioade<br>constant, independent de N (mărimea grupului) |
| Încărcarea nodului | constantă pe perioada T<br>$< 8L^*$ pentru pierdere de pach. de 15% (ridicată) |
| Rata FP | reglabilă (via K)<br>se reduce exponențial odată cu creșterea K |
| Completness | în O(log(N)) cu probabilitate ridicată<br>limitată în timp |

# SWIM failure detection (5)

- $PM(T)$ este exponențială în $-K$; ea depinde și de $p_{ml}$ și $p_{failure}$
- vezi Gupta, I., Chandra, T.D., Goldszmidt, G.S., "On Scalable and Efficient Distributed Failure Detectors"

# Timpul de detecție așteptat (1)

- toate nodurile au nodul defect în lista de membership
- toate trimit ping-uri aleator
- probabilitatea ca nodul defect să fie ping-uit de un anume alt nod: $1/N$
- probabilitatea ca nodul defect să nu fie ping-uit de acel anume alt nod: $1 - 1/N$
- probabilitatea ca nodul defect să nu fie ping-uit de nici unul din celelalte $N - 1$ noduri: $\left(1 - \frac{1}{N}\right)^{N-1}$
- probabilitatea ca nodul defect să fie ping-uit de măcar unul din celelalte $N - 1$ noduri: $1 - \left(1 - \frac{1}{N}\right)^{N-1}$

# Timpul de detecție așteptat (2)

- probab. ca nodul defect să fie ping-uit în timpul T': $1 - \left(1 - \frac{1}{N}\right)^{N-1} = 1 - e^{-1}$
- (construcție similară cu limita e, vezi `http://aleph0.clarku.edu/~djoyce/ma122/elimit.pdf`)
- timpul așteptat pentru prima detecție T: $E[T] = T' \cdot \frac{e}{e-1}$
- ex. dacă probab. de ping e de 0.5 în 2 secunde, val. așteptată este de de 4 secunde, adică $E[T] = \frac{1}{p} \cdot T'$
- Completeness: toate nodurile alive detectează failure-ul nodului căzut
  - în cele din urmă
  - în cel mai defavorabil caz în O(N) perioade T

# Timpul de detecție așteptat (3)

- în cazul cel mai nefavorabil, este posibil ca nodul căzut să fie detectat foarte târziu (pentru că nimeni nu face ping la el din cauza randomizării)
- reducerea timpului de detecție pentru toate nodurile, la O(N):
  - se selectează fiecare nod ca ping target, la traversarea listei de membership
  - lista este ordonată aleator; se face round-robin ping
  - permutare random a listei după fiecare traversare
- astfel, fiecare failure este detectată în cel mult 2N-1 perioade
- păstrează proprietățile de detecție a failures (FP și scalabilitatea)

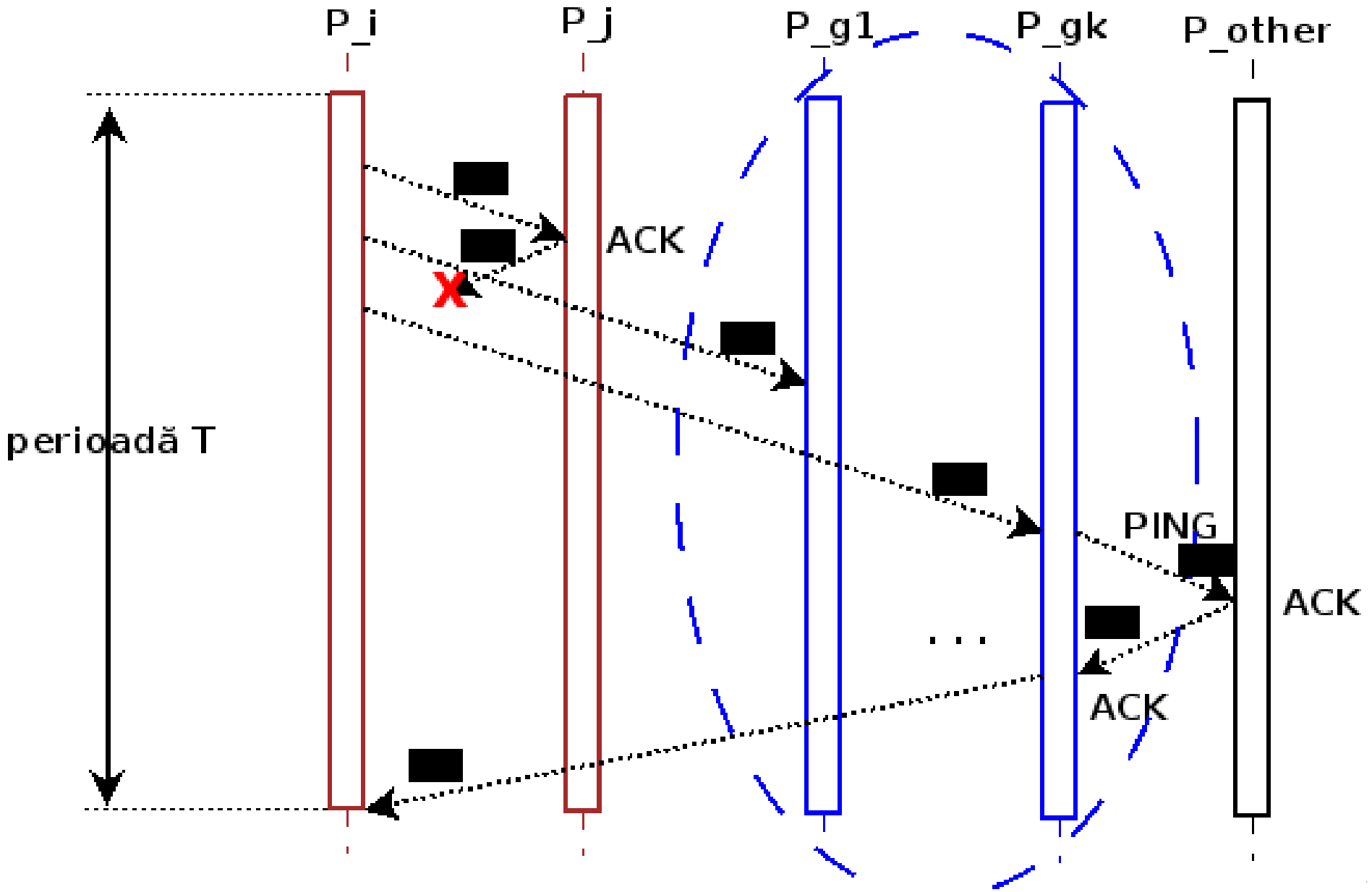
# Dissemination (1)

- componenta de failure detection este separată de componenta de diseminare
- componenta de **diseminare** realizează distribuirea listei de membership către nodurile grupului
- ea poate trimite și informații de JOIN și LEAVE voluntare
- trebuie să trimită această informație către TOȚI membrii grupului

# Dissemination (2)

- multicast IP (hardware IP)
  - unreliable
  - multicast-uri simultane multiple
- point-to-point (TCP / UDP)
  - costisitoare, pe măsură ce numărul de noduri din grup crește
- **piggy-back** pe mesajele de tip failure detection
  - nu sunt alte mesaje adăugate
  - diseminare de tip infection (de tip SWIM)
  - fiecare mesaj PING, respectiv ACK, va purta și lista de membership cunoscută de nodul de la care pleacă mesajul
  - la primirea unui mesaj PING sau ACK, nodul își va actualiza lista de membership
  - crește încărcarea per nod la O(N log(N))

# Dissemination (3)



# Dissemination de tip infection

- diseminare de tip epidemie
- după $\lambda$ log(N) perioade, $N^{-(2\lambda-2)}$ procese nu au primit update
- fiecare nod menține un buffer cu nodurile joined / evacuate
- piggy-back din acest buffer
- se preferă update-urile recente
- elementele buffer-ului sunt șterse după o vreme (cele mai vechi)
- se șterg înregistrări mai vechi ca $\lambda$ log(N) perioade
- $\lambda$ determină consistency
- dimensiunea buffer-ului este constantă $\rightarrow$ încărcarea per nod devine O(log(N))

# Scăderea ratei FP

- mecanismul de scădere a ratei FP-urilor
- pot apărea totuși FP datorită:
  - noduri perturbate
  - pierderi de pachete, de ex. congestii
- ping-ul indirect poate să nu rezolve problema (în zona nodului ping-uit, pierderile de mesaje pot fi corelate)
- ideea: de a **suspecta** un nod înainte de a-l declara failed către grup

# Automat de stare

- se folosește o mașină de stare cu 3 stări pentru nodul din listă, { ALIVE, SUSPECTED, FAILED }
- ALIVE $\rightarrow$ SUSPECTED: când failure detector realizează că un nod s-ar putea să nu fie ALIVE
- SUSPECTED $\rightarrow$ ALIVE: la primirea unui mesaj informativ despre acel nod, mai nou decât evidența curentă
- SUSPECTED $\rightarrow$ FAILED: la expirarea unui timeout ($t_{cleanup}$)
- componenta de diseminare va trimite toate nodurile împreună cu starea lor, { ALIVE, SUSPECTED sau FAILED }
- putem avea o ciclare succesivă între ALIVE și SUSPECTED

# Incarnation number

- cum se distinge între suspiciuni multiple și se previne ciclarea între ALIVE și SUSPECTED:
  - se menține un **incarnation number**, pentru fiecare nod
  - incarnation number-ul pentru nodul (procesul) $p_i$ poate fi incrementat numai de $p_i$, atunci când primește un mesaj (Suspect, $p_i$)
  - la primirea acestui mesaj (Suspect, $p_i$), $p_i$ va incrementa incarnation number și va începe diseminarea informației că de fapt este ALIVE
- incarnation number-ul mai mare suprascrie numărul mai mic, deodată cu starea
- pentru același incarnation number, (SUSPECT, i.n.) $>$ (ALIVE, i.n.), și informația existentă de tip ALIVE este suprascrisă
- un mesaj (FAILED, i.n.) va suprascrie orice informație, indiferent de incarnation number suprascris
- un nod FAILED, dacă este ALIVE, va trebui să facă din nou join

# Final

- Failure-urile sunt regula, nu excepția în DC, deoarece avem de-a face cu un număr mare de noduri
- fiecare sistem distribuit folosește un failure detector
- multe sisteme distribuite folosesc un serviciu de membership

- detecția de tip ring failure este folosită pe IBM SP2 și pe clustere de dimensiuni mici spre medii

- detecția de tip gossip este presupusă că funcționează pe AWS EC2/S3