Synchronizacja w systemach rozproszonych

Klasyfikacja

- *Sieciowe SO*
- *Rozproszone SO*
- *Systemy wieloprocesorowe z podziałem czasu*

Sieciowe SO

Sieć – moduł komunikacyjny – sieciowy SO – lokalny SO – procesy użytkownika

- *luźno pow. oprogramowanie i l. p. sprzęt*
- *Praca lokalna lub rlogin, telnet, ssh*
- *Zdalne kopiowanie (rcp, ftp, scp)*
- *Każda maszyna niezależna*

Prawdziwy rozproszony SO

- *Ściśle powiązane oprogramowanie, luźno powiązany sprzęt*
- *Wrażenie jednolitego systemu (wirtualny monoprocesor)*
- *Jeden SO*
- *Dostęp do zasobów jednolity (lokalne i globalne)*
- *SO nadzoruje przemieszczanie*
  - *danych (FTP; NFS, SMB)*
  - *obliczeń (RPC; komunikaty)*
  - *procesów*
- *Komputery nie autonomiczne*
- *Amoeba, Mach, Chorus*

Systemy wieloprocesorowe z podziałem czasu

- *Ściśle powiązane oprogramowanie, ściśle powiązany sprzęt (nie ma pamięci lokalnej)*
- *Jedna kolejka uruchomień (lista procesów do wykonania)*

- ***Systemy scentralizowane*** *(wspólna pamięć, zegar) --- sekcja krytyczna; synchronizacja – semafory, monitory,..*

- ***Systemy rozproszone*** *– synchronizacja, następstwo zdarzeń*
  - *Rozwiązanie scentralizowane – inf. dot. synchronizacji w 1 m-cu + proces zarządzający*
    - *Duże obciążenie*
    - *Utrata niezawodności*

Właściwości systemów rozproszonych

- *Informacje rozmieszczone na wielu maszynach*
- *Procesy podejmują decyzje na pdst. inf. lok.*
- *Unikanie skupienia w 1 m-cu elementów wrażliwych na awarie*
- *Brak wspólnego zegara (globalnego czasu)*
  - *Problem - uzgadnianie czasu (synchronizacja wszystkich zegarów?)*
    - *(np. make)*

Zegary logiczne

- *Każdy komputer – własny timer (kryształ)*
- *Wiele JC – odchylenie czasu*

Synchronizacja zegarów – Lamport

- *Nie jest istotne, aby wszystkie procesy uzgadniały dokładnie wartość czasu*
- *Ważna jest kolejność zdarzeń*
- *Zegary logiczne*
- *Zegary fizyczne*

Synchronizacja zegarów logicznych - Algorytm Lamporta

- *Relacja uprzedniości zdarzeń* $a \rightarrow b$
  - *Zachodzi gdy:*
    - a,b *w tym samym procesie i a przed b*
    - a,b *w różnych procesach a- wysłanie komunikatu, b – odebranie komunikatu*
  - *Przechodnia*
  - *Jeśli* x, y *w różnych procesach i nie wymieniają komunikatów*

    $\sim(x \rightarrow y)$ *i* $\sim(y \rightarrow x)$ *– zdarzenia współbieżne (*concurrent*)*
- *Sposób pomiaru czasu* - $C(a)$

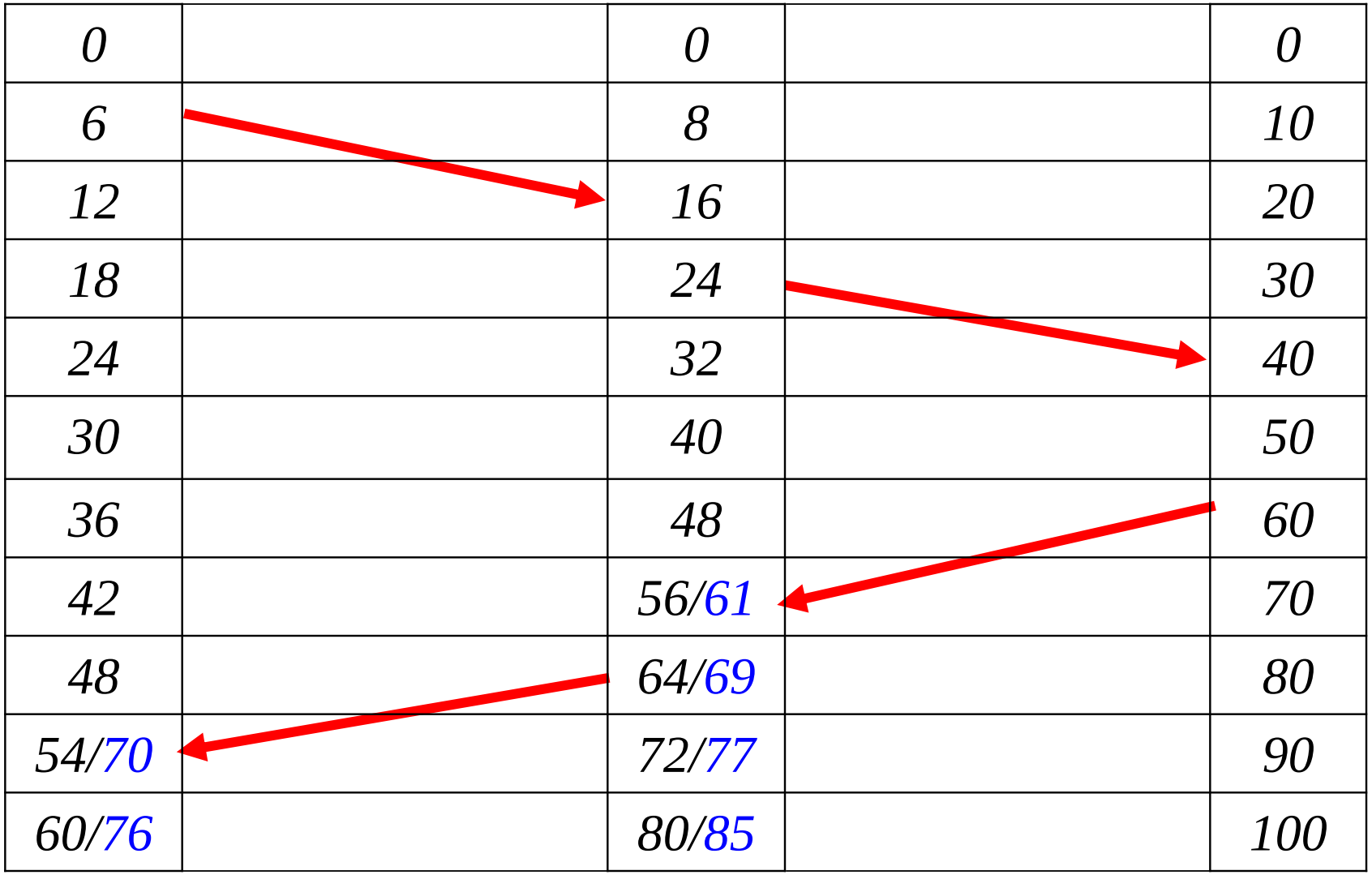
$$a \rightarrow b \Leftrightarrow C(a) < C(b)$$

  - C *– czas zegarowy – płynie do przodu*

Algorytm Lamporta

- *3 procesy; 3 maszyny; 3 zegary o różnych częstotliwościach*
- *Komunikat zawiera czas swojego nadania*
- *Czas przesyłania komunikatu >=1*

- *Jeśli czas przybycia komunikatu <= czas nadania komunikatu => odbiorca przesuwa czas zegara na wartość = czas nadania + 1*
  - *Dodatkowy warunek: żadne 2 zdarzenia nie powinny mieć tego samego czasu (20 → 20.1; 20.2)*


0
0
0
6
8
10
12
16
20
18
24
30
24
32
40
30
40
50
36
48
60
42
56/61
70
48
64/69
80
54/70
72/77
90
60/76
80/85
100

- *AL. zapewnia całkowite uporządkowanie wszystkich zadań w systemie rozproszonym*

  - *a,b w tym samym procesie i a przed b => C(a) < C(b)*
  - *a,b w różnych procesach a- wysłanie komunikatu, b – odebranie komunikatu => C(a) < C(b)*
  - *dla wszystkich a,b: C(a) ≠ C(b)*

ZEGARY LOGICZNE

Zegary fizyczne

- *Systemy czasu rzeczywistego*
- *Potrzeba zewnętrznego zwielokrotnionego zegara fizycznego*
- *Problemy:*
  - *Synchronizacja z zegarem rzeczywistym*
  - *Synchronizacja między sobą*
- *Pomiar czasu*
  - *Astronomowie*
  - *Fizycy*
- *BIH w Paryżu ->TAI*
- *UTC (Universal Coordinate Time) -> nadajnik WWV-Colorado; MSF – Anglia; GEOS*

Algorytmy synchronizacji zegarów

Zał. *każda maszyna ma swój czasomierz (H przerwań /sec)*

max. wsp. odchylenia – ρ

Cel *różnica zegarów <=δ*

resynchronizacja programowa co δ/(2ρ) sec

Algorytmy:

scentralizowane: Cristiana (pasywny serwer czasu);

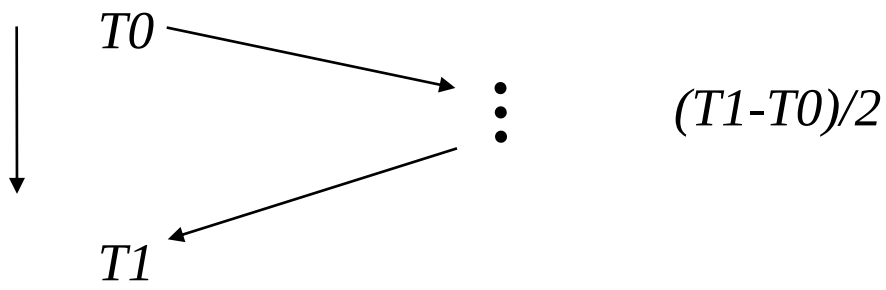
z Berkeley (aktywny serwer czasu);

zdecentralizowany: uśredniania

Algorytm Cristiana

Serwer czasu - ma odbiornik WWV

- *Każda maszyna wysyła co $\delta/(2\rho)$ sec kom. z pytaniem o bież. czas*
- *Nadawca ustawia czas na C_{UTC} + czas przes. kom.*
- *Problem:*
  - *- C_{UTC} < czas nadawcy*
    - *Stopniowa zmiana czasu*
  - *Oszacowanie czasu przesyłania komunikatu*



Algorytm z Berkeley

Brak odbiornika WWV

- *Serwer czasu odpytuje maszyny*
- *Oblicza średni czas*
- *Wszystkie maszyny wyrównują czasy*

Algorytm uśredniania

Przedziały resynchronizacji stałej długości

$[\ T0 + iR;\ T0 + (i+1)R\ ]$

Każda maszyna:

- *na pocz. każdego przedziału ogłasza czas swojego zegara*
- *zbiera ogłoszenia od innych*
- *oblicza wartość nowego czasu (np. wart. średnia po odjęciu skrajnych wartości)*

WZAJEMNE WYŁĄCZANIE

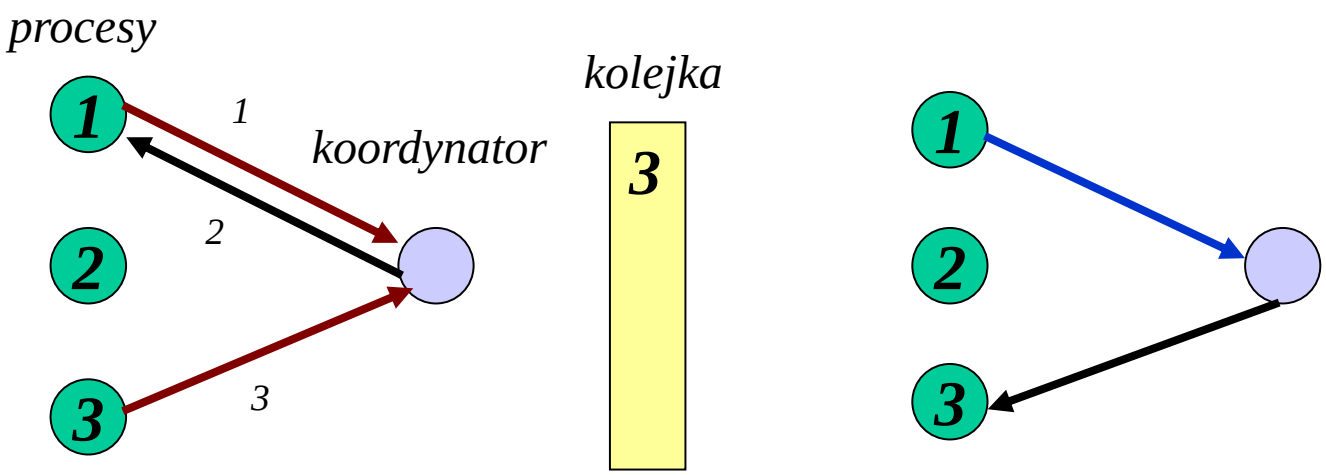
- *Algorytm scentralizowany*
- *Algorytm rozproszony*
- *Algorytm pierścienia z żetonem*

Algorytm scentralizowany

Wyróżniony proces – koordynator *(np. najwyższy adr. siec.)*

- *Proces wysyła do koordynatora zamówienie we. do SK*
- *Jeśli SK wolna koordynator odsyła odp. OK*
  - *po jej nadejściu proces wchodzi do SK*
  - *po wy. z SK wysyła do koordynatora komunikat*
- *Jeśli SK zajęta koordynator nie odpowiada;*
  - *proces -> do kolejki*
  - *koordynator odpowiada po zwolnieniu SK*

Algorytm scentralizowany



komunikaty:

zamówienie

pozwolenie

zwolnienie

Algorytm scentralizowany

- *Zapewnia wzajemne wyłączanie*
- *Reguła sprawiedliwa: brak głodzenia, we. do SK w kolejności zamawiania*

- *Wady:*
  - *Koordynator - wąskie gardło;*
    wrażliwość na uszkodzenie całego systemu

Algorytm rozproszony
Ricarta i Agrawali

- *Bazuje na alg. Lamporta – wymaga uporządkowania zdarzeń*

Proces, który chce we. do SK *tworzy komunikat:*

- *nazwa sekcji*
- *swój numer*
- *bieżący czas*

i wysyła go do wszystkich procesów (komunikat potwierdzany)

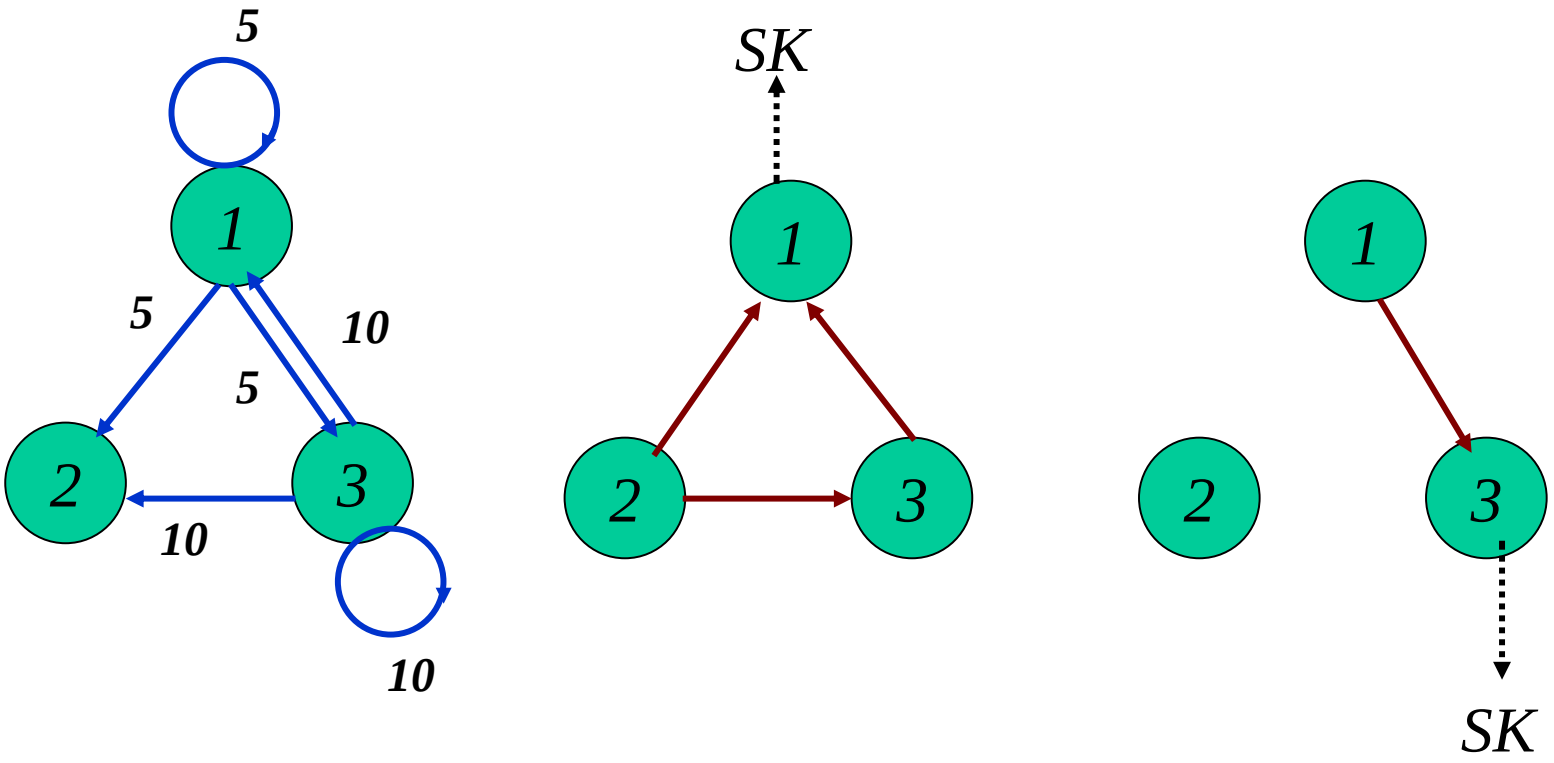
odbiorca

- *Jeśli nie jest w SK i nie chce tam wchodzić odsyła komunikat OK*
- *Jeśli jest w SK nie odpowiada; ustawia zamówienie w kolejce*
- *Jeśli chce we. do SK porównuje znacznik czasu komunikatu odebranego z wysłanym przez siebie; jeśli własny jest większy – odsyła OK.; jeśli mniejszy – ustawia zamówienie w kolejce*

Proces, który chce we. do SK

- *Po skompletowaniu wszystkich pozwoleń (OK) wchodzi do SK*
- *Po wy z SK wysyła komunikat OK do wszystkich procesów z kolejki i usuwa je z kolejki*

Algorytm rozproszony



zamówienie

pozwolenie -OK

Algorytm rozproszony

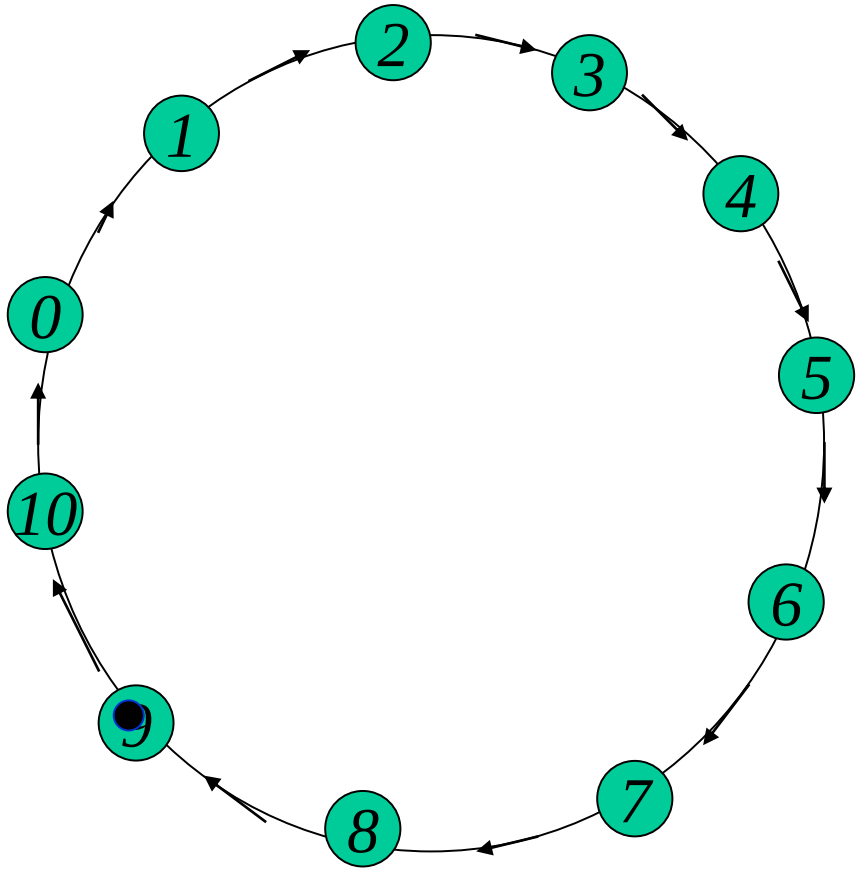
problemy

- *Uszkodzenie jednego procesu – blokuje wszystkie próby wejścia do SK*
- *Duży ruch w sieci; każdy proces zaangażowany we wszystkie decyzje o wejściu do SK*
- *Próba naprawy:*
  - *Po nadejściu zamówienia odbiorca odsyła odpowiedź (pozwolenie lub odmowa)*

Wzajemne wyłączanie bez blokad i zagłodzenia

wolniejszy, kosztowniejszy, mniej odporny, ale możliwy

Algorytm pierścienia z żetonem



procesy nieuporządkowane-> logiczny pierścień

żeton – komunikat k->(k+1)mod n

1 żeton – 1 SK

- *brak głodzenia*

PROBLEMY:

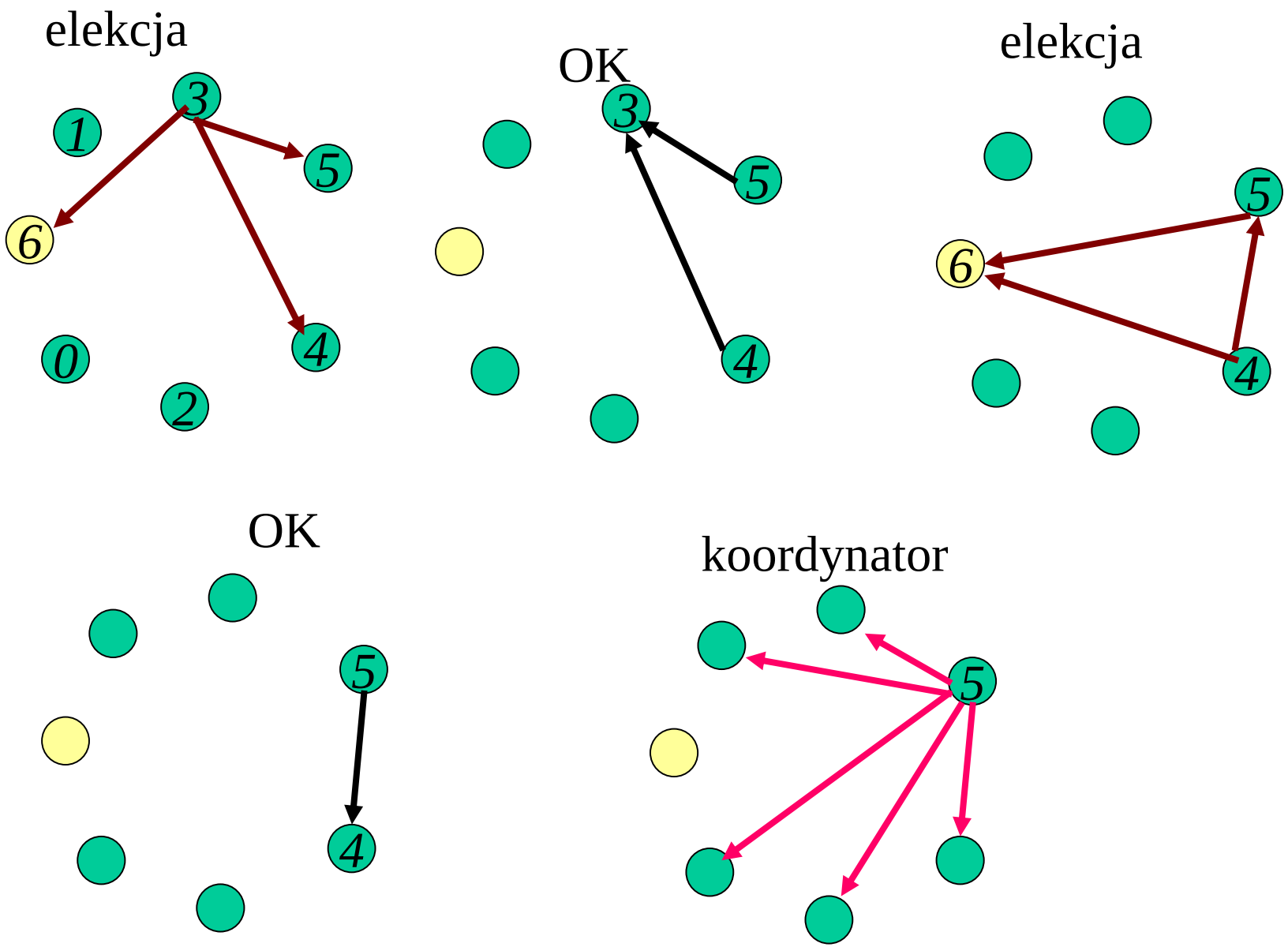
- *zaginięcie żetonu*
- *komunikaty wysyłane bez wzgl. na potrzebę*

porównanie

algorytm	*l. kom. we. do SK*	*opóźnienie we. do SK*	*problemy awarie*
scentralizowany	*3*	*2*	*koordynator*
rozproszony	*2(n-1)*	*2(n-1)*	*dowolny proces*
pierścienia z żetonem	*1..?*	*0..n-1*	*dowolny proces, żeton*

Wybór koordynatora elekcja

- ***Algorytm tyrana***
  - *Proces **P** o numerze **n** rozpoczyna elekcję*
  - *Wysyła komunikat **ELEKCJA** do procesów o numerach > n*
  - *Jeśli nikt nie odpowiada – P – koordynatorem*
  - *Jeśli odpowie proces o numerze > n - przejmuje kontrolę*
  - *Zwycięzca wysyła komunikat **KOORDYNATOR***


elekcja
1
3
5
6
4
0
2
OK
3
5
4
elekcja
5
6
4
OK
5
4
koordynator
5

Wybór koordynatora algorytm pierścieniowy

- *Pierścień bez żetonu*
- *Proces tworzy komunikat ELEKCJA + numer procesu wysyła go do następnego działającego*
- *Każdy następny proces dodaje swój numer*
- *Gdy komunikat wraca do nadawcy zamiana typu komunikatu – KOODYNATOR (najwyższy numer)*
- *Po kolejnym obiegu komunikat znika*
-

Transakcje niepodzielone

- *Wyższy poziom abstrakcji*
- *Zasada „wszystko albo nic”*
  - *inicjatywa jednego procesu;*
  - *wspólne działania;*
  - *zatwierdzenie przez wszystkie procesy -> utrwalenie wyników*
  - *brak zatwierdzenia -> powrót do pierwotnego stanu*
- *Właściwości transakcji:*
  - *Niepodzielność (*atomicity)
  - *Spójność (*consistence)
  - *Izolacja (*isolation)
  - *Trwałość* (duarability)
- *Transakcje zagnieżdżone (trwałość odnosi się do najwyższego poziomu)*
- *Implementacja*
  - *Prywatna przestrzeń robocza*
  - *Rejestr zapisów wyprzedzających*

Prywatna przestrzeń robocza

- *PPR zawiera wszystkie pliki i obiekty, do których proces ma dostęp*
- *Duży koszt -> modyfikacje:*
  - *Kopie jedynie zmienianych plików (wskaźniki do przestrzeni roboczej procesu rodzicielskiego)*
  - *Do przestrzeni roboczej – kopie indeksów (i-węzły) modyfikowanych plików; nowe bloki (cienie)-> modyfikacja indeksów*

    zaniechanie transakcji –> zwolnienie prywatnych bloków

    zatwierdzenie transakcji -> przemieszczenie prywatnych indeksów w przestrzeń roboczą procesu macierzystego

Rejestr zapisów wyprzedzających lista zamiarów

- *Przed każdą zmianą bloku zapis rekordu do RZW:*
  - *Transakcja*
  - *Adres (plik, blok) zmieniany*
  - *Stara wartość*
  - *Nowa wartość*
- *Po zatwierdzeniu transakcji – rekord zatwierdzenia do RZW*
- *W przypadku zaniechania – przywrócenie stanu pierwotnego na pdst. RZW (wycofanie* rollback*)*