# Sisnkronisasi Sistem Paralel dan Terdistribusi

## Overview

- Sistem tersebar sebenarnya adalah proses-proses yang berkolaborasi atau bekerjasama. Sebelumya telah dibahas komunikasi yang merupakan dasar kesemuanya, dibahas juga penamaan yang penting untuk sumber daya berbagi, dan sekarang akan dibahas issue sinkronisasi.
- Sikronisasi sendiri diperlukan baik di sistem tunggal atau sistem terdistribusi dengan alasan yang sama.

## Tujuan

1. Mahasiswa memahami apa itu sinkronisasi dan pentingnya sinkronisasi pada sistem terdistribusi.
2. Mahasiswa juga mengetahui secara singkat teknik-teknik sinkronisasi disertai kondisi-kondisi untuk menerapkan teknik-teknik tersebut .

# Definisi

- Sinkronisasi adalah proses pengaturan jalannya beberapa proses pada saat yang bersamaan.
- Tujuan utama sinkronisasi adalah menghindari terjadinya inkonsitensi data karena pengaksesan oleh beberapa proses yang berbeda (mutual exclusion) serta untuk mengatur urutan jalannya proses-proses sehingga dapat berjalan dengan lancar dan terhindar dari deadlock dan starvation.
- Sinkronisasi umumnya dilakukan dengan bantuan perangkat sinkronisasi. Penyelesaian terhadap masalah ini sangat penting karena perkembangan teknologi sistem komputer menuju ke sistem *multiprocessing*, terdistribusi dan paralel yang mengharuskan adanya proses-proses kongkuren.

# Sinkronisasi Clock

- Algoritma untuk sinkronisasi dalam sistem terdistribusi memiliki beberapa sifat:
  1. Informasi yang relevan tersebar di beberapa computer
  2. Keputusan pembuatan proses hanya berdasarkan informasi local.
  3. Peristiwa kegagalan dengan penyebab tunggal di dalam sistem harus dihindarkan
  4. Tidak tersedianya clock atau sumber waktu global yang akurat.

- Sikronisasi merupakan bagian penting untuk kerjasama dalam :
  - Pemakaian sumberdaya berbagi (*Sharing resources*)
  - Pengurutan kejadian
  - Kesepakatan clock tersebar

## Contoh Tidak Adanya Kesepakatan Clock Global


- Gambar ini menggambarkan bahwa bila waktu pada output o adalah 2144, Kemudian source codenya dimodifikasi di komputer lain yang clocknya lebih lambat, sehingga waktu source code adalah 2143.
- Karena source code memiliki waktu yang lebih lama daripada file objeknya, maka make tidak akan melakukan rekompilasi.

## Sinkronisasi Straightforward

- Cara yang paling mudah untuk menentukan waktu adalah dengan bertanya langsung ke server waktu (*Universal Coordinated Time* - UTC), hanya saja akan banyak perbedaan dalam *request* .
- Karena waktu merupakan dasar dari cara orang berpikir, dan akibat tidak adanya sinkronisasi clock juga sangat dramatis, seperti yang dilihat pada contoh sebelumnya, sehingga wajar saja bila dalam pembahasan sikronisasi dimulai dengan pertanyaan sederhana :
  - Mungkinkan mensinkronkan semua clock yang ada dalam sistem tersebar ?

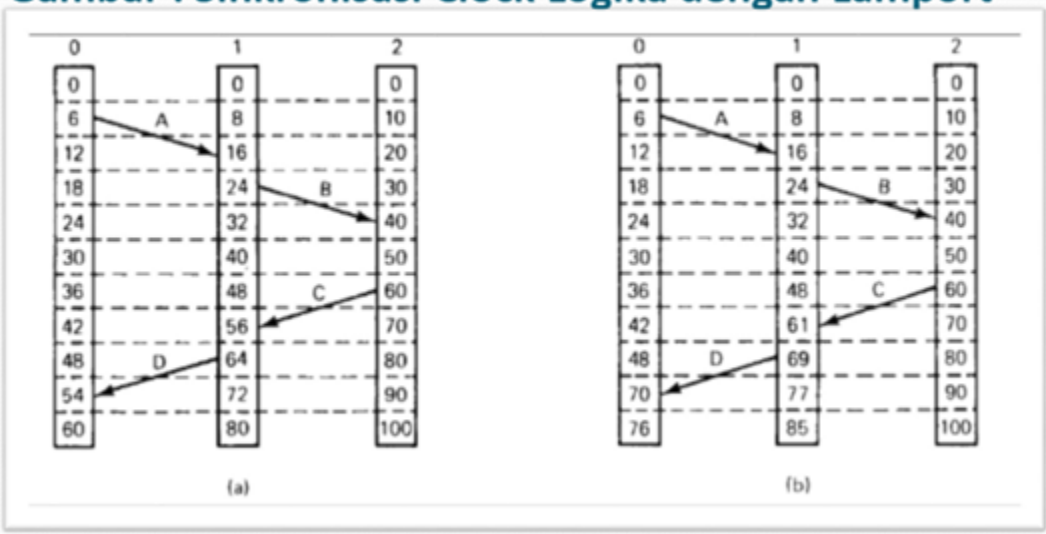
## 1.1. Clock logika

- Boleh dikatakan semua komputer memiliki rangkaian pencatat waktu. Walaupun penggunakan kata Clock sudah meluas, kata yang lebih tepat adalah timer untuk merujuk komponen dari rangkaian tersebut.
- Timer ini menggunakan crystal quartz sebagai sumber frekuensinya. Walaupun frekuensi osilator pada osilator kristal biasanya stabil, tetap saja tidak mungkin menjamin bahwa semua kristal yang bekerja diberbagai komputer memiliki frekuensi yang persis sama.
- Selalu ada sedikit perbedaan yang terjadi dan mengakibatkan perbedaan waktu pula yang disebut **clock skew**.
- Berbagai algoritma telah dikembangkan untuk menangani sinkronisasi clock dan beberapanya akan dibahas berikut ini.

- **Algoritma Lamport**
  - Menurut Lamport, sikronisasi clock tidak harus dilakukan dengan nilai mutlak clocknya, karena yang diperlukan dalam sikronisasi proses-proses adalah urutan proses tersebut.
  - Jadi yang dipentingkan adalah konsistensi internal clock, bukan apakah clock tersebut harus sama persis dengan waktu real.
  - Clock jenis ini biasanya disebut clock logika.

- **Pengurutan peristiwa**
  - Sejumlah problem yang penting akan terpecahkan bila pegurutan peristiwa yang jelas dapat dibuat, bahkan bila waktu realnya tidak diketahui. Untuk mensinkronkan clock logika Lamport mendefinisikan relasi yang disebut **happened-before.**
  - Ekspresi a-> b dibaca "a terjadi sebelum b" dan artinya semua proses sepakat bahwa kejadian pertama adalah a, diikuti sesudahnya kejadian b. Relasi happen-before dapat diamati langsung dalam dua situasi:
  - Bila a dan b adalah kejadian (event) dalam proses yang sama, dan a terjadi sebelum b, maka a-> b adalah true.
  - Bila a adalah kejadian dari sebuah pesan yang dikirim oleh sebuah proses,dan b adalah kejadian dimana pesan tersebut diterima oleh proses lain,maka a-> b adalah true juga.

## Gambar : Sinkronisasi Clock Logika dengan Lamport

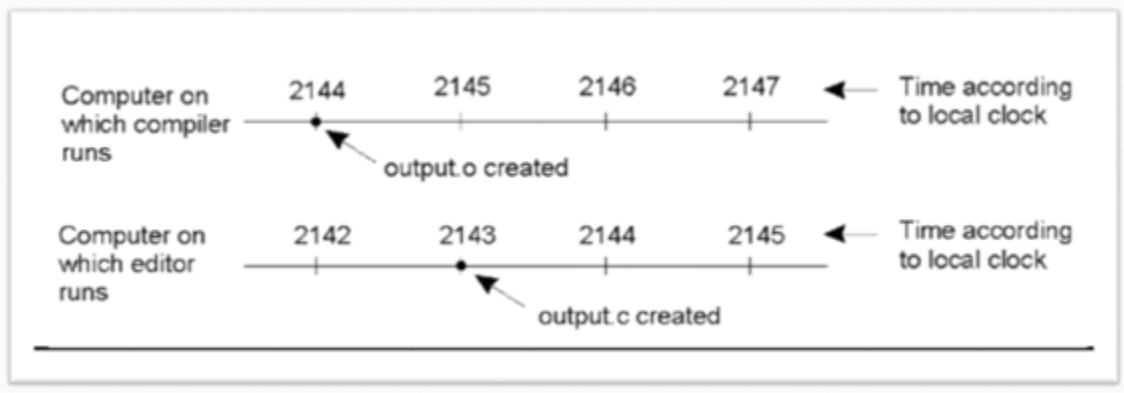



- Bila a adalah kejadian dari sebuah pesan yang dikirim oleh sebuah proses,dan b adalah kejadian dimana pesan tersebut diterima oleh proses lain, maka a-> b adalah true juga.
- Pada gambar (a) tampak tiga buah sistem dengan clock Masing – masing yang bekerja dengan laju yang berbeda, dan gambar (b) clock sistem dikoreksi dengan algoritma Lamport.
- Cara untuk menetapkan waktu ke semua kejadian dalam sistem tersebar tergantung pada kondisi berikut:
  1. Bila a terjadi sebelum b di proses yang sama , C(a) < C(b).
  2. Bila a dan b mewakili kejadian pengiriman dan penerimaan pesan, maka C(a) < C(b)
  3. Untuk semua kejadian a dan b, C(a) <> C(b)

## 1.2. Clock fisik

- Pada beberapa sistem, waktu clock aktual menjadi penting, contohnya real – time sistem. Untuk sistem ini diperlukan clock fisik eksternal. Karena alasan efisiensi dan redundansi, clock fisik jamak biasanya digunakan, yang mengakibatkan ada dua masalah muncul:
  - Bagaimana mensinkronkan eksternal clock tersebut dengan clock sebenarnya
  - Bagaimana mensinkronkan antar clock yang ada.

- Sebelum membahas jawaban masalah di atas, terlebih dahulu dilihat bagaimana pengukuran waktu aktual dilakukan.
- Saat dimana matahari mencapai titik tertinggi di langit disebut **transit of the sun**, dan terjadi di siang hari. Interval antar dua transit berturutan disebut solar day. Sedangkan solar second didefinisikan tepat 1/86400 dari solar day.
- International Atomic Time (disingkat IAT) adalah rata-rata jumlah tick dari jam atom cesium 133 sejak tanggal 1 januari 1958 dibagi 9.192.631.770.
- Disebabkan waktu siang bertambah lama, TAI menjadi lebih lambat dibanding solar second. Untuk mengkoreksinya, digunakan leap second dengan cara meloncati waktu TAI sehingga sama dengan solar second (lihat gambar). Waktu yang telah dikoreksi ini disebut Universal Coordinated Time UTC).
- NIST memiliki beberapa stasiun radio gelombang pendek yang memancarkan pulsa pada setiap awal detik UTC, yang dapat digunakan untuk sinkronisasi. Stasiun ini dikenal dengan nama WWV.




## 1.3. Algoritma Sinkronisasi Clock

- Frekuensi tick clock logika tergantung dari nilai yang dimuat ke counter. Nilai ini yang menentukan resolusi clock. Interval waktu yang lebih kecil dari resolusi tidak dapat dibedakan.
- Laju pergeseran clock adalah perubahan offset antara clock dengan nominal referensi ideal per unit waktu yang diukur di referensi.
- Clock hardware hanya berupa nilai di dalam register, seperti nilai 32 bit, yang kelak di Roll-over. Penanganan dilakukan dengan mengubah konstanta yang ditambahkan untuk memperoleh clock software yang biasanya berkisar di orde mikrodetik atau milidetik dari tanggal yang disepakati.

1. **Algoritma Cristian**

   Bila sebuah mesin memiliki penerima WWV sehingga dapat berfungsi sebagai time server. Secara periodik, setiap mesin mengirim pesan ke time server menanyakan waktu terkini, Cutc.

2. **Algoritma Berkeley**

   Algoritma Berkeley digunakan untuk mensinkronkan clock relatif terhadap clock lainnya, dan bukan terhadap master clock tertentu.

3. **Algoritma Rata-rata**

   Berbeda dengan metode sebelumnya yang terpusat, maka metode ini mensinkronkan clock dengan cara desentralisasi. Cara kerjanya dengan membagi waktu ke dalam interval resinkronisasi yang lebarnya tetap.

4. **Sumber Clock Eksternal Jamak**

   Algoritma ini menjadi dasar untuk protokol NTP (**Network Time Protocol**). Interval waktu dapat ditentukan dengan menggunakan algoritma Cristian dengan waktu tunda perambatan yang diketahui. Digunakan dalam sistem yang disinkronkan dengan sangat akurat.

## 1.4 Penggunaan Clock Sinkron

- Pelaksanaan sinkronisasi clock dalam skala luas terjadi baru -baru ini saja, yang salah satu teknologi enabling - nya adalah internet.
- Adalah mungkin mensinkronkan jutaan clock dalam orde milidetik dengan UTC.
- Berbagai algoritma baru yang menggunakan clock sinkron mulai bermunculan, berikut ini contohnya.

  1. **At-Most-Once Message Delivery**

     Setiap pesan membawa pengenal koneksi dan timestamp. Untuk setiap koneksi, server menyimpan timestamp terbaru ke dalam tabel. Bila ada pesan masuk dengan timestamp yang lebih lama daripada Timestamp yang disimpan, maka pesan tersebut akan ditolak dan dianggap sebagai duplikat.

  2. **Konsistensi Cache Berbasis Clock**

     Konsistensi cache dalam file System tersebar menjadi perhatian karena setiap client menginginkan cache file di lokal komputer. Bila dua komputer memodifikasi file secara bersamaan, berpotensi menyebabkan inkonsistensi.

## 2.1 Algoritma Terpusat

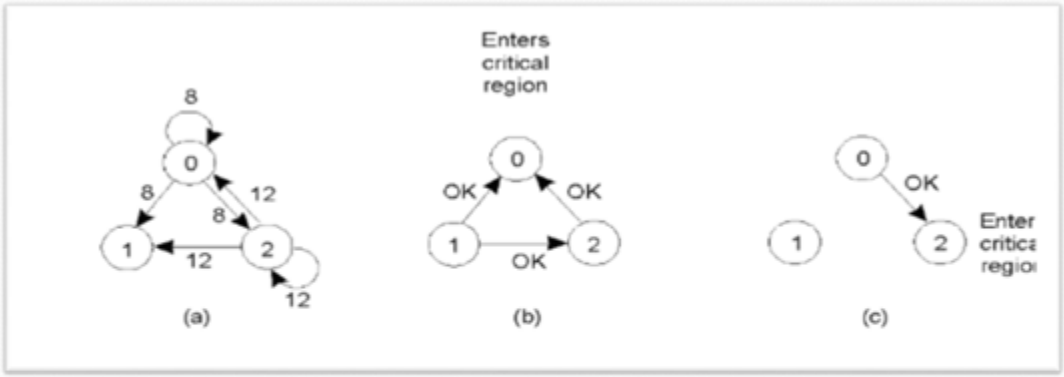
- Pada algoritma terpusat, kondisi mutual exclusion (mutex) ditangani oleh sebuah proses yang dipilih sebagai koordinator untuk mengatur entry ke critical region. Setiap proses yang ingin meminta mutex mengirim pesan request ke koordinator. Bila proses tersebut menerima pesan reply dari koordinator maka proses tersebut diijinkan masuk ke daerah kritis. Sesudah keluar dari daerah kritis, proses mengirim pesan release ke koordinator dan melanjutkan eksekusinya.


a. Proses 1 meminta ijin (request) ke koordinator untuk masuk ke critical region. Ijin diberikan (grant).
b. Proses 2 meminta ijin ke koordinator untuk masuk ke critical region yang sama. Koordinator tidak menjawab.
c. Bila proses 1 keluar dari critical region, proses tersebut memberitahu (release) koordinator yang kemudian mengijinkan proses 2.

## 2.2 Algoritma Tersebar

- Kejadian kegagalan karena penyebab tunggal tidak dapat ditoleransi dalam sistem tersebar, sehingga para peneliti mengembangkan berbagai algoritma mutual exclusion tersebar.
- Algoritma ini bekerja dengan membuat sebuah proses yang ingin memasuki daerah kritis , terlebih dulu membuat pesan yang berisi nama daerah kritis yang ingin dimasuki, nomor proses dan waktu terkininya.
- Pesan ini dikirim ke semua proses dengan asumsi komunikasi yang digunakan reliable

- Bila sebuah proses menerima pesan request dari proses yang lain, respon yang diberikan tergantung dari state proses terhadap nama daerah kritis yang dalam pesan tersebut. Ada tiga kasus penerima yang mungkin yaitu
  a. Bila penerima tidak berada dalam daerah kritis dan tidak ingin masuk, maka pesan Ok dikirim balik.
  b. Bila penerima sudah berada di dalam daerah kritis, maka tidak ada pesan yang dikirim.
  c. Bila penerima ingin masuk ke daerah kritis tapi belum masuk, maka proses ini akan membandingkan catatan waktu dari pesan masuk dengan pesan yang dikirimkan. Bila pesan masuk memiliki catatan lebih lama, penerima akan membalas dengan pesan OK. Sebaliknya bila pesannya sendiri memiliki catatan waktu yang lebih lama maka penerima akan meletakkan pesan masuk ke antrian dan tidak membalas apapun.



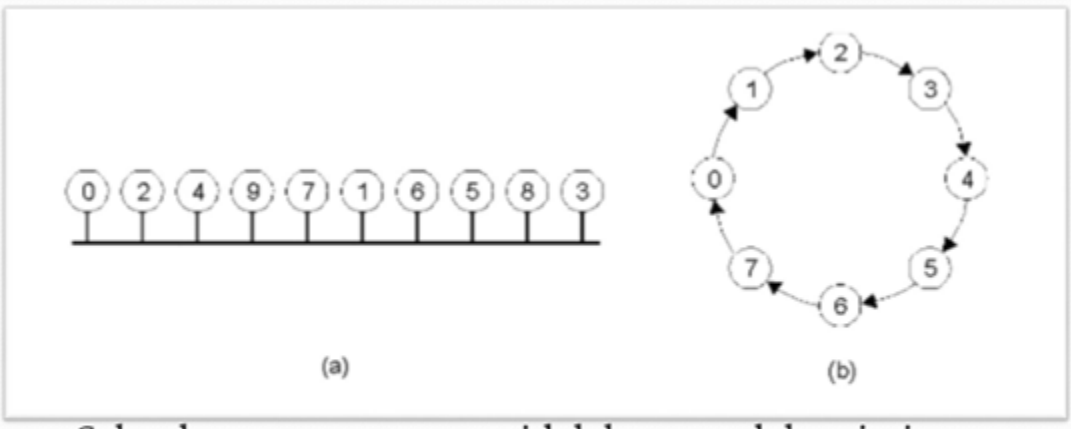

a. Dua proses ingin masuk ke daerah kritis yang sama pada waktu yang bersamaan pula.

b. Proses 0 memiliki timestamp yang lebih lama sehingga proses 0 menang.

c. Bila proses 0 selesai, pesan OK dikirim sehingga proses 2 sekarang dapat masuk ke daerah kritis.

## 2.3 Algoritma Token Ring

- Disini digunakan sebuah jaringan bus dengan proses - proses yang tidak berurutan.
- Melalui perangkat lunak, ring logika disusun dengan setiap proses ditetapkan posisinya di dalam ring seperti pada gambar b. Posisi ring dapat dialokasikan dengan menggunakan urutan nomor alamat jaringan atau dengan cara lain.
- Hal yang terpenting adalah setiap proses harus tahu siapa proses sesudahnya.




a. Sebuah grup proses yang tidak berurut dalam jaringan.
b. Ring logika yang disusun dalam software perbandingan Tiga Algoritma

## 2.4 Perbandingan Tiga Algoritma

- Algoritma terpusat paling mudah dan efisien dibanding kedua algoritma lainnya. Hanya tiga proses yang dibutuhkan untuk masuk dan keluar dari daerah kritis: Request , grant dan release .
- Algoritma tersebar paling sensitif terhadap kejadian crash.

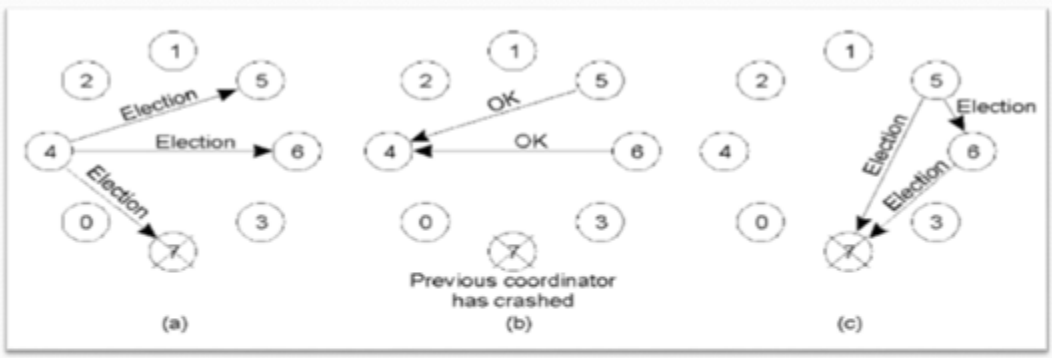
| **Algoritma** | **Pesan *per entry/exit*** | ***Delay sebelum entry (in message times)*** | ***Problem*** |
|---|---|---|---|
| Terpusat | 3 | 2 | Koordinator *crash* |
| Tersebar | 2 (n-1) | 2 (n-1) | Proses *crash* |
| Token Ring | 1 to infinity | 0 to n-1 | Token hilang,proses *crash* |

# Algoritma Pemilihan

- Banyak algoritma tersebar membutuhkan sebuah proses yang berfungsi sebagai koordinator, inisiator, sekuenser, atau pelaksana fungsi khusus lain. Beberapa contoh seperti koordinator pada algoritma mutual exclusion terpusat.
- Bila koordinator tersebut mengalami kegagalan karena hostnya down, sistem harus dapat melanjutkan eksekusi hanya dengan memulai lagi sebuah copy proses koordinator baru di host yang lain. Algoritma yang menentukan dimana copy koordinator baru tersebut harus dimulai lagi disebut algoritma pemilihan.
- Ada dua algoritma pemilihan yang akan dibahas untuk dua jenis konfigurasi sistem tersebar.

## 2.4.1 Algoritma Bully

- Bila sebuah proses mendapatkan koordinator tidak lagi menanggapi request yang dikirim, maka proses pemilihan akan diinisiasi. Proses P mengadakan pemilihan sebagai berikut:
  1. P mengirim pesan ELECTION ke semua proses dengan nomor proses yang lebih besar.
  2. Bila tidak ada tanggapan, proses P memenangkan pemilihan ini dan menjadi koordinator.
  3. Namun bila salah satu proses dengan nomor yang lebih tinggi menjawab, proses tersebutlah yang akan mengambil alih proses pemilihan. Pekerjaan proses P sendiri selesai disini.

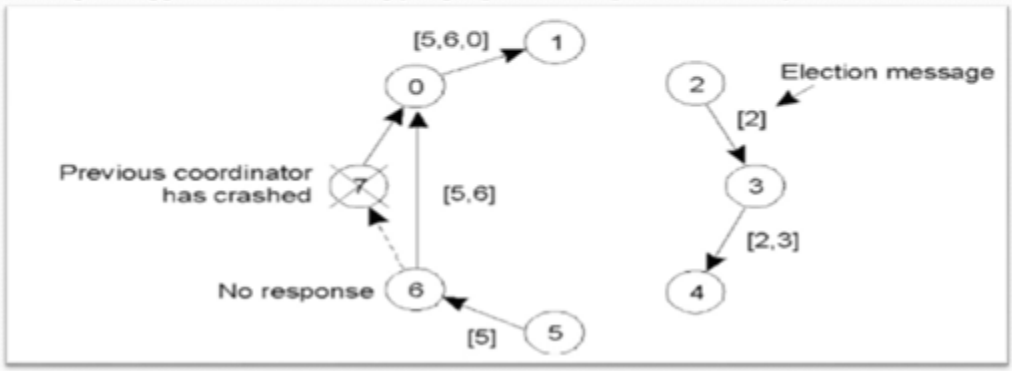



- Dalam gambar proses pemilihan dengan algoritma bully dapat dilihat sebagai berikut
  - Proses 4 mengadakan pemilihan (ELECTION)
  - Proses 5 dan 6 merespon, memberitahu 4 untuk berhenti
  - Sekarang 5 dan 6 masing-masing akan mengadakan pemilihan

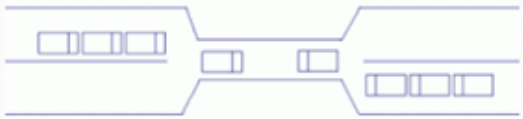
## 2.4.2 Algoritma Ring

- Algoritma ini berbasis ring tanpa token, dengan ersyaratan bahwa setiap proses harus sudah berurutan baik secara logika ataupun fisik.
- Bila sebuah proses mendapatkan koordinatornya tidak berfungsi, maka pesan ELECTION yang berisi nomor prosesnya dikirim ke proses berikut yang lebih tinggi nomornya.

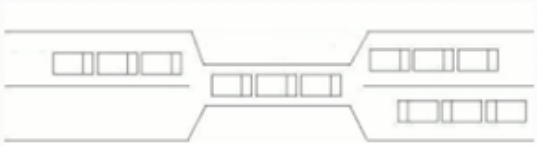
- Dalam gambar terlihat bahwa proses 2 dan 5 mendapatkan proses 7 yang berperan sebagai koordinator mengalamicrash.
- Kemudian proses 2 dan 5 membangun pesan ELECTION dan memulai sirkulasi pesan ini.
- Akhirnya pesan tersebut akan tersebar ke segala arah, kemudian kedua proses 2 dan 5 akan mengubah pesan tersebut menjadi pesan COORDINATOR yang disirkulasikan lagi, dengan anggota dan urutan ring yang tepat sama seperti sebelumnya.



- **Deadlock** adalah suatu kondisi dimana terdapat dua proses atau bahkan lebih dalam antrian proses yang lain untuk melepaskan resource yang sedang dipakai



- **Starvation** adalah kondisi yang biasanya terjadi setelah *deadlock*



Selesai