Sistem Operasi

Pertemuan 5 – Konkurensi: Mutual Exclusion & Synchronization

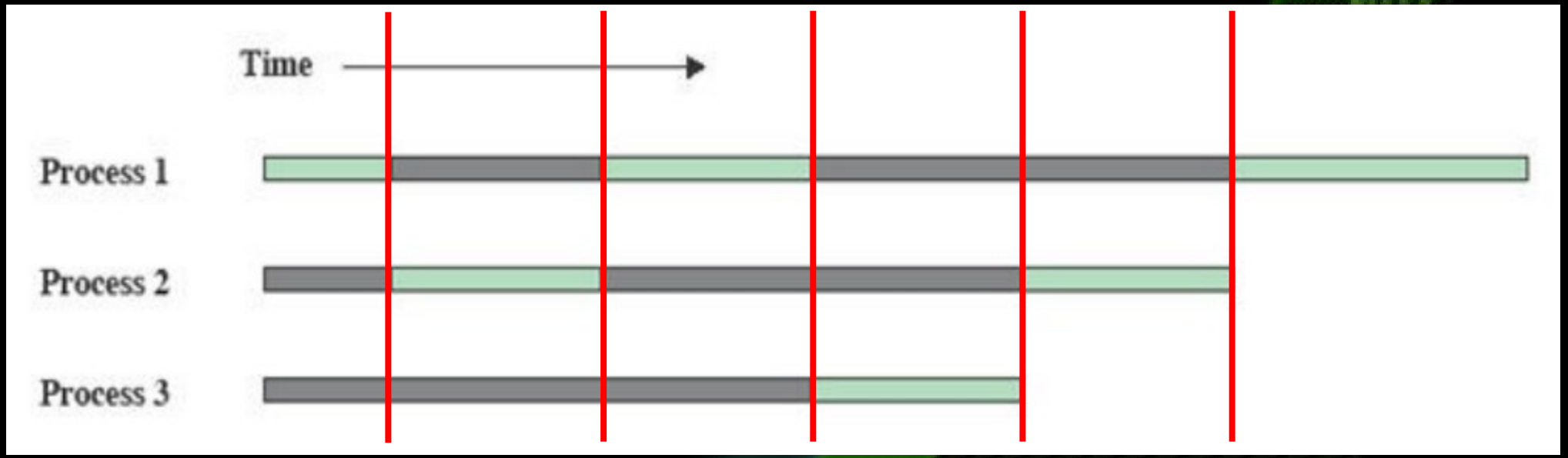
Konkurensi

- Konkurensi
- Mutual Exclusion
- Syncronization

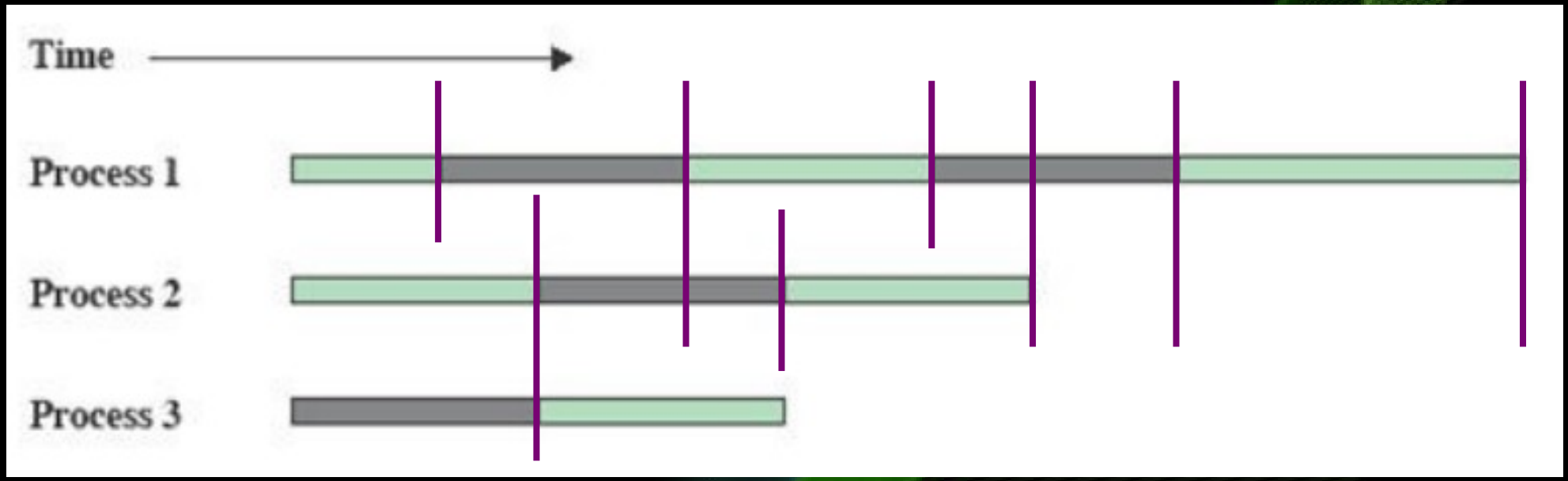
Konkurensi

- Definisi:
  - Kemampuan sistem opreasi untuk mengeksekusi banyak intruksi dalam waktu bersamaan.
- Selain di sistem operasi, konkurensi juga dijadikan filosofi pemrograman untuk memecahkan kompleksitas program

Multiprogramming Uniproc



Overlapping Multiproc



Kesulitan Konkurensi

- Sharing sumber daya global
  - Penulisan suatu shared variable: urutan penulisan sangat penting
  - Masalah besar adalah penulisan tidak lengkap
- Pengelolaan alokasi resource secara optimal
  - Sulit menemukan error pemrograman karena hasilnya bersifat tidak deterministic and reproducible.

Contoh Multiprocessor

```
void echo()
{
   chin = getchar();
   chout = chin;
   putchar(chout);
}
```

Lanjutan

```
Proses P1
        .
chin = getchar();
        .
chout = chin;
putchar(chout);
        .
```

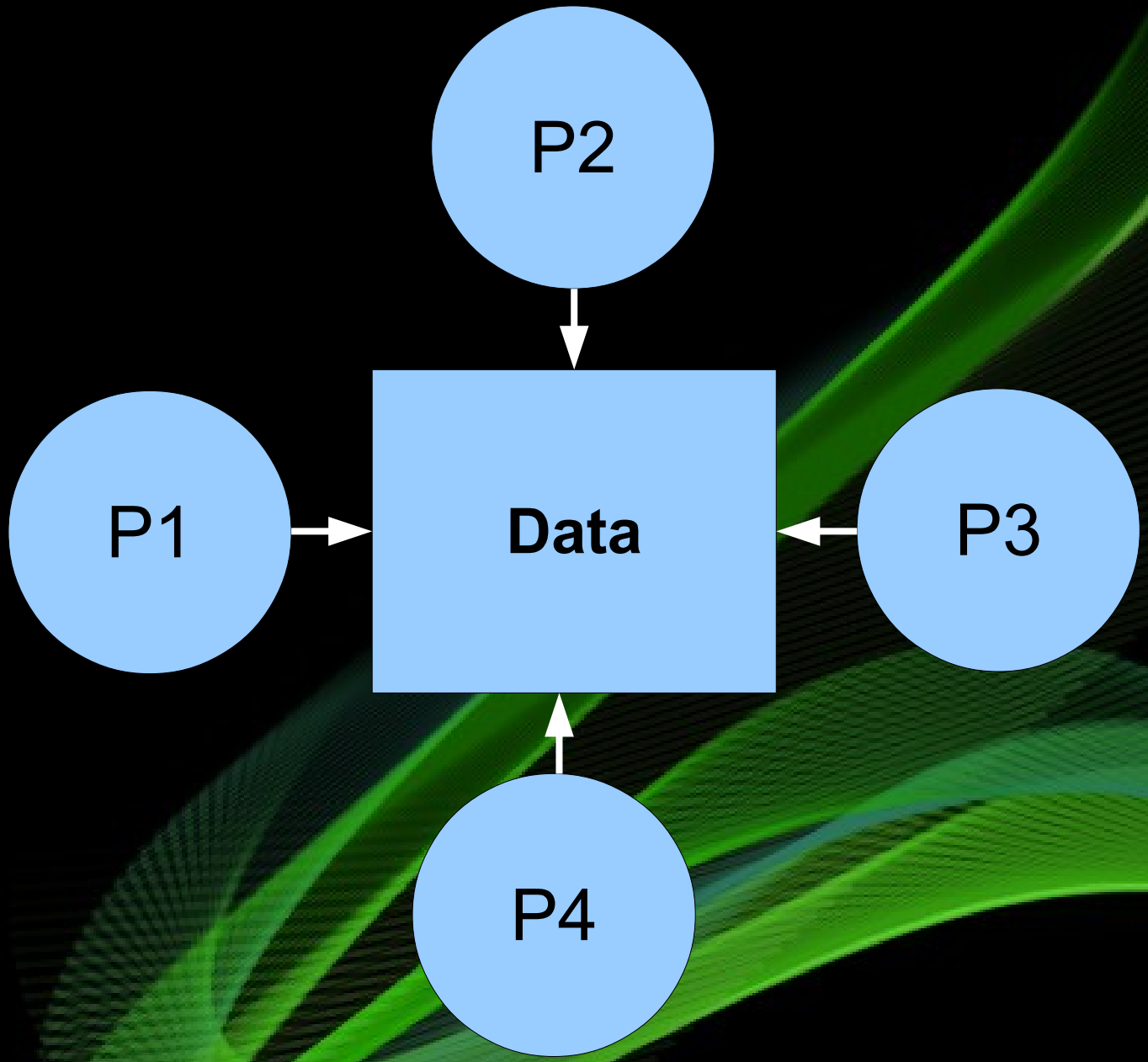
```
Proses P2
        .
        .
chin = getchar();
chout = chin;
                .
putchar(chout);
```

Jika pengaksesan I/O hanya boleh tunggal

Kondisi *Race*

- Race condition terjadi ketika
  - Banyak proses atau thread membaca & menulis item data
  - Hasil akhir dari aktifitas baca & tulis tersebut tergantung pada urutan eksekusi dari proses yang terlibat.
- Output tergantung pada siapa yang terakhir menyelesaikan race

Ilustrasi



SO dan Konkurensi

- SO harus
  - Menjaga track dari berbagai proses
  - Meng-alokasi-kan dan men-dealokasi-kan sumber daya
  - Melindungi data & resource dari gangguan proses lain
  - Memastikan bahwa proses & output terbebas dari kecepatan pemrosesan

Kompetisi Antar Proses

- Ada tiga masalah kendali utama:
  - Kebutuhan Mutual Exclusion
    - Critical section (bagian kritis dari proses)
    - Deadlock
    - Starvation

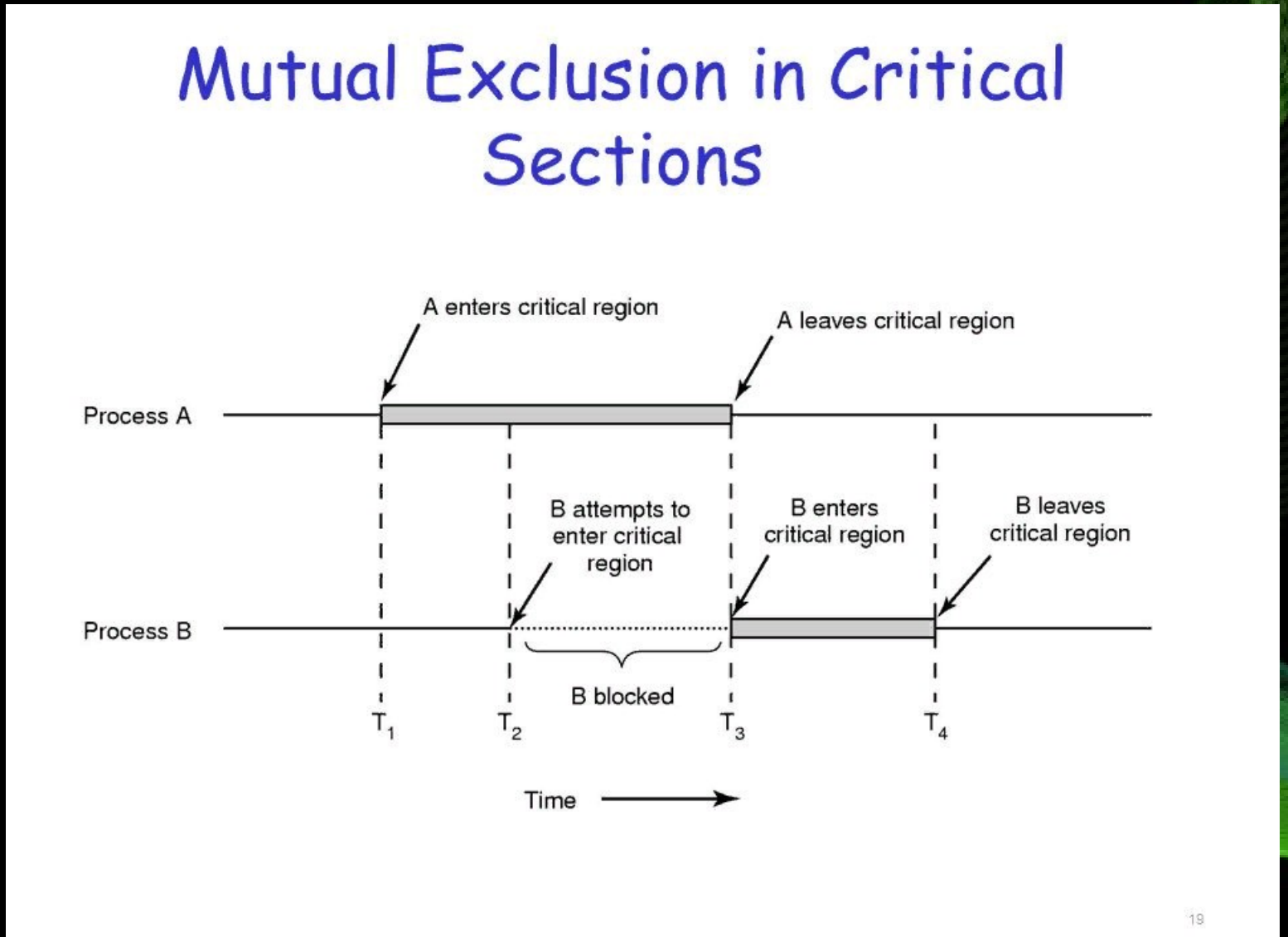
Mutual Exclusion

- Uniprosesor hanya boleh melakukan interleaving
- Mematikan Interupsi
  - Proses berjalan sampai ia meng-invoke suatu layanan SO atau sampai ia diinterupsi
  - Disabling interrupts menjamin terwujudnya mutual exclusion
  - Tidak akan bekerja pada arsitektur multiprocessor

Syarat Mutual Exclusion

- Proses harus tidak didelay akses ke suatu critical section saat tidak ada proses lain yang menggunakannya
- Tidak ada asumsi mengenai kecepatan proses relatif atau jumlah proses
- Proses tetap di dalam critical section-nyahanya selama waktu terbatas tertentu (finite)

Ilustrasi


Mutual Exclusion in Critical Sections
A enters critical region
A leaves critical region
Process A
B attempts to enter critical region
B enters critical region
B leaves critical region
Process B
B blocked
T1
T2
T3
T4
Time
19

Keuntungan Mutual Exclusion

- Dapat diterapkan terhadap banyak proses pada processor tunggal atau multi processor yang berbagi (sharing) main memory
- Dapat digunakan untuk mendukung banyak critical section

Kerugian Mutual Exclusion

- **Busy-waiting** mengkonsumsi waktu processor
- **Starvation** mungkin ketika suatu proses meninggalkan critical section dan lebih dari satu proses menunggu (waiting).
  - Beberapa proses dapat ditolak aksenya dalam waktu tak terbatas.
- Kemungkinan Deadlock

Synchronization

- Dalam dunia IT, sinkronisasi dibagi menjadi 2
  - Sinkronisasi Proses
    - Yang di mana beberapa proses bergabung menjadi satu kesatuan untuk meraih suatu aksi berurutan
  - Sinkronisasi Data
    - Sebuah konsep untuk membuat data cadangan yang valid demi integritas

Ilustrasi

Synchronization

- Synchronisation dicapai dengan variabel kondisi dalam suatu monitor
  - Hanya dapat diakses oleh monitor.
- Fungsi monitor:
  - Cwait(c): Men-suspend eksekusi dari proses yang memanggil pada kondisi c
  - Csignal(c) Me-resume eksekusi dari beberapa proses yang diblok setelah cwait pada kondisi yang sama

Lanjutan

- Komunikasi memerlukan sinkronisasi
  - Sender (pengirim) harus mengirim sebelum receiver (penerima) dapat menerima
- Apa yang terjadi terhadap proses setelah ia menjalankan primitif send atau receive?
  - Sender & receiver mungkin (bisa pula tidak) menjadi blocking (waiting for message)

Blocking Send & Receive

- Sender & receiver diblok sampai message tersampaikan (delivered)
- Dikenal sebagai rendezvous
- Memungkinkan sinkronisasi ketat (tight) antar proses.

Interaksi Proses

- Ketika proses berinteraksi satu dengan lainnya, dua syarat fundamental harus terpenuhi:
  - Sinkronisasi, dan
  - komunikasi.
- Message Passing adalah (satu) solusi untuk syarat kedua
  - Bonus: bekerja dengan shared memory & dengan sistem terdistribusi.

Message Passing

- Fungsi aktual dari message passing normalnya disediakan dalam bentuk pasangan primitif:
  - send (destination, message)
  - receive (source, message)

Addressing Pesan

- Proses pengiriman perlu mampu menentukan proses mana yang sebaiknya menerima message
  - Direct addressing (langsung)
  - Indirect Addressing (tidak langsung)

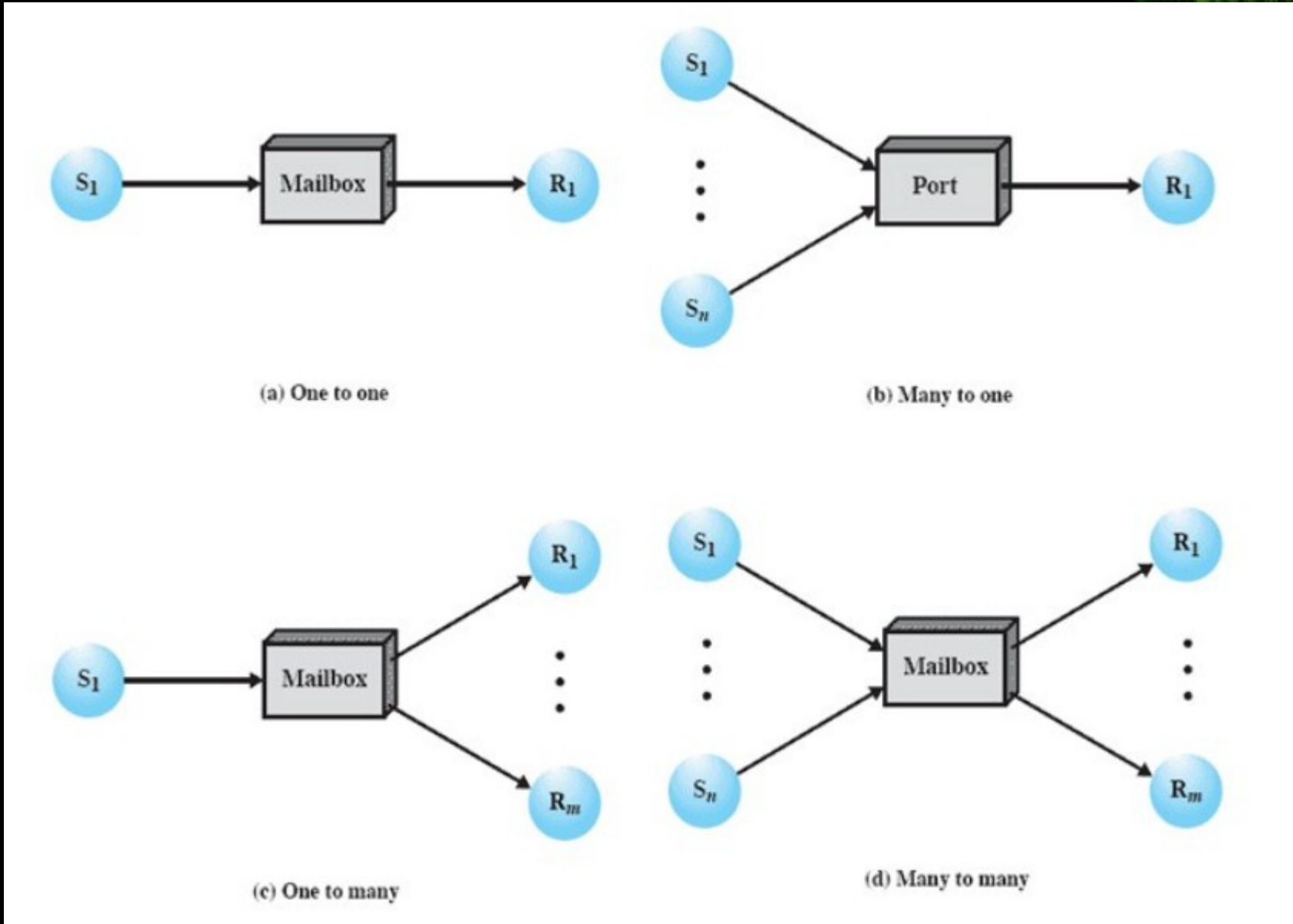
Direct Addressing

- Primitif send menyertakan suatu pengenal (identifier) khusus dari proses tujuan
- Primitif receive segera dapat mengetahui proses mana yang menjadi tujuan message
- Primitif receive dapat memanfaatkan parameter source untuk mengembalikan suatu nilai ketika operasi receive selesai dikerjakan.

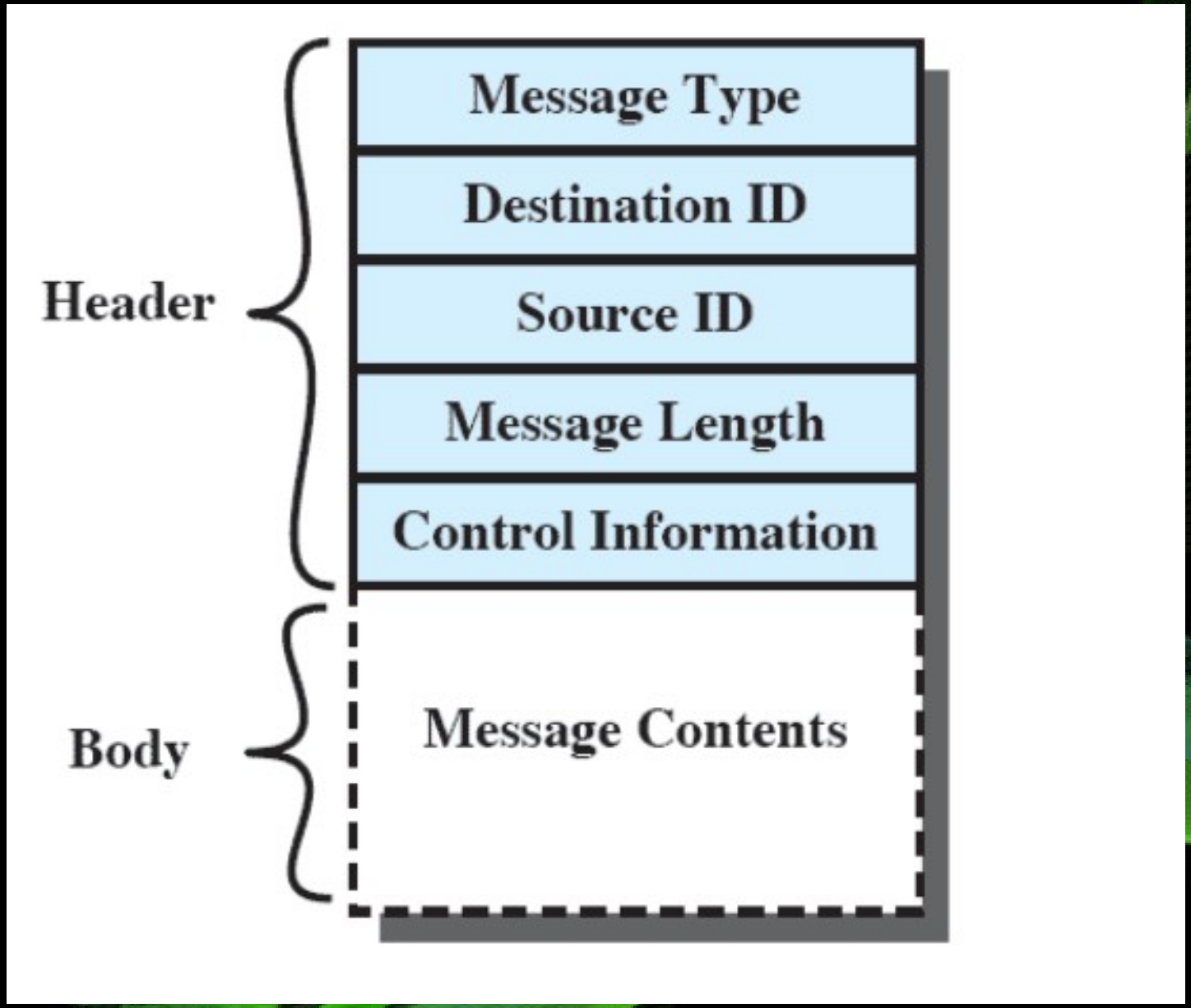
Indirect Addressing

- Message dikirim ke suatu struktur shared data yang mengandung antrian (queues)
- Queues disebut pula mailboxes
- Satu proses mengirimkan suatu message ke mailbox & proses lain mengambil message dari mailbox tersebut

Ilustrasi


S1
Mailbox
R1
(a) One to one
S1
Sn
Port
R1
(b) Many to one
S1
Mailbox
R1
Rm
(c) One to many
S1
Sn
Mailbox
R1
Rm
(d) Many to many

Format Message



Masalah R/W

- Suatu area data dishare antar banyak proses
- Beberapa proses hanya membaca area data, beberapa hanya menulis ke area tersebut. Kondisi untuk dicapai:
  - Banyak readers boleh membaca file at once.
  - Hanya satu writer pada satu waktu yang dapat menulis
  - Jika suatu writer sedang menulis ke file, maka tidak ada reader yang dapat membacanya