Lecture 7: Deadlock and Starvation

Summary

1. Computer system overview (Chapter 1)
2. Basic of virtual memory; i.e. segmentation and paging (Chapter 7 and part of 8)
3. Process (Chapter 3)
4. Mutual Exclusion and Synchronization (Chapter 5 section 1-4)
    - Conditions for race avoidance.
    - Strict alternation.
    - Semaphores.
    - Producers and Consumers problem.
    - Hardware support for mutual exclusion.
    - Monitors.
5. Threading (user mode and kernel mode).

Deadlock

1. Deadlock: Permanent blocking of a set of processes that either compete for system resources or communicate with each other.
    - A set of processes is deadlocked when each process in the set is blocked awaiting an event that can only be triggered by another blocked process in the set.
        - Permanent.
        - No efficient solution.
2. Examples:
    - Traffic deadlock (see figure 6.1 in the book)
    - Processes and computing resources (see joint progress diagrams, Figure 6.2 and Figure 6.3).
3. Reusable resources: Used by one process at a time and not depleted by that use. Processes obtain resources that they later release for reuse by other processes.
    - Examples: CPUs, printers, I/O channels, main and secondary memory, files, databases, and semaphores (see Figure 6.4).
    - Embedded systems example (see Figure 6.5):
4. Consumable resources: Created (produced) and destroyed (consumed) by a process (see Figure 6.6).
    - Examples: Interrupts, signals, messages, and information in I/O buffers.
    - Deadlock may occur if a Receive message is blocking.
    - May take a rare combination of events to cause deadlock.

```
Assume 200KB of available physical memory.

===============
P1
...
Request 80K bytes;
...
Request 60K bytes;
===============
P2
...
Request 70K bytes;
...
Request 80K bytes;
===============
```

Figure 6.5: Deadlock in a memory constrained device.

```
Assume 200KB of available physical memory.

===============
P1
...
Receive (P2);
...
Send (P2, M1);
===============
P2
...
Receive (P1);
...
Send (P1, M2);
===============
```

Figure 6.6: Deadlock with blocking I/O.

5. Conditions for Deadlock (all for must be present for deadlock to occur):
   - Mutual Exclusion.
   - Hold and Wait.
   - No preemption.
   - Circular wait.
6. Deadlock prevention: tries to prevent any of the four conditions from occurring.
7. Deadlock avoidance: dynamically decides whether the current resource allocation request will, if granted, potentially lead to a deadlock.
   - Do not start a process if its demands might lead to deadlock.

  - Know the total amount of resources.
  - Know the amount of available resources.
  - Know the exact amount of needed resources by a process.
  - A process is only started if the maximum claim of all current processes plus those of the new process can be met.
- Do not grant an incremental resource request to a process if this allocation might lead to deadlock.
  - Banker's algorithm: assume fixed number of processes and fixed number of resources (see accompanying slides).
    * State: claim matrix, allocation matrix, resource vector, available vector.
    * Safe state: there is at least one sequence of resource allocations to processes that does not result in a deadlock.
    * Unsafe state: a state that can would lead to deadlock if the request is granted.
- Restrictions for deadlock avoidance:
  - Maximum resource requirement must be stated in advance.
  - Processes under consideration must be independent; no synchronization requirements.
  - There must be a fixed number of resources to allocate.
  - No process may exit while holding resources.

8. Deadlock detection: use existing allocation matrix (A), resource vector, available vector and the new request metrix (Q) (see accompanying slides).

9. Recovering from Deadlock

   - Abort all deadlocked processes.
   - Back up each deadlocked process to some previously defined checkpoint, and restart all processes (original deadlock may occur).
   - Successively abort deadlocked processes until deadlock no longer exists.
   - Successively preempt resources until deadlock no longer exists.

10. Dining philosophers (see Figure 6.11).

    - Come up with some solutions.

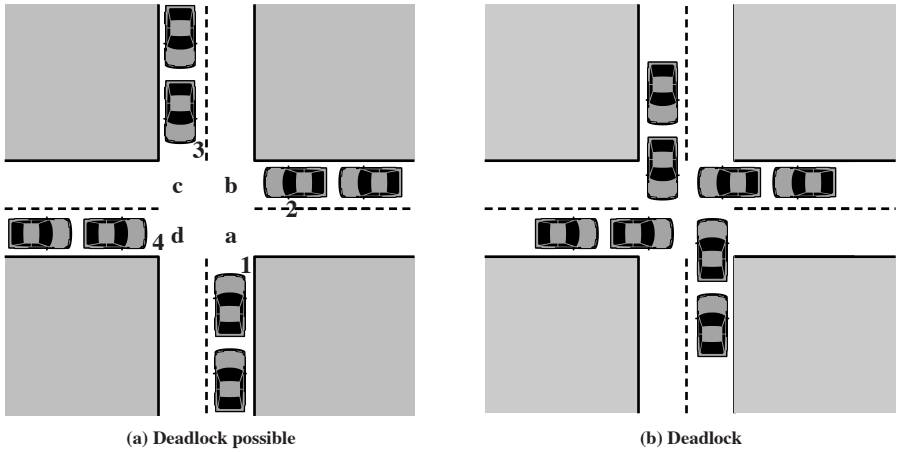

Figure 6.1 Illustration of Deadlock

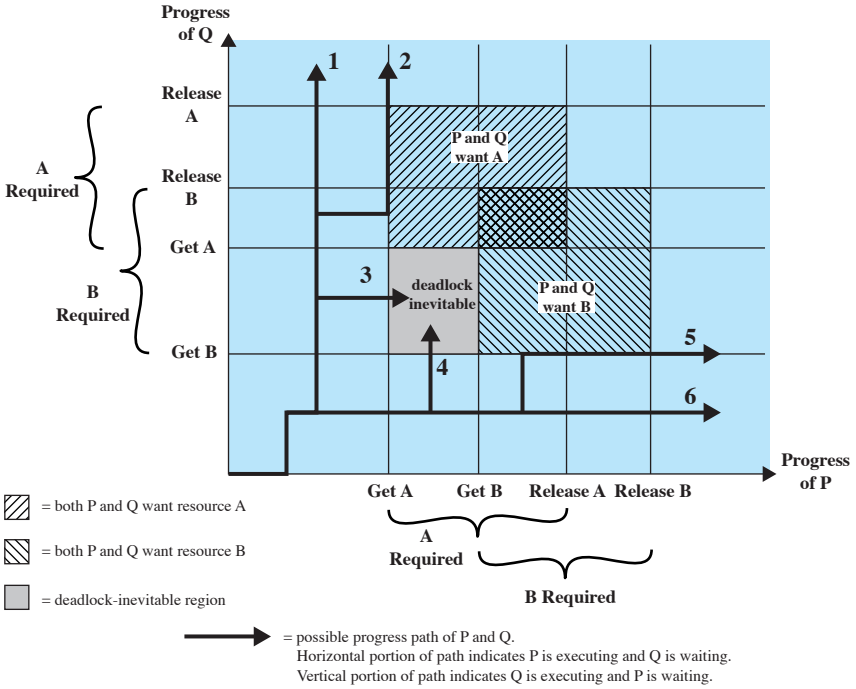

Figure 6.2 Example of Deadlock

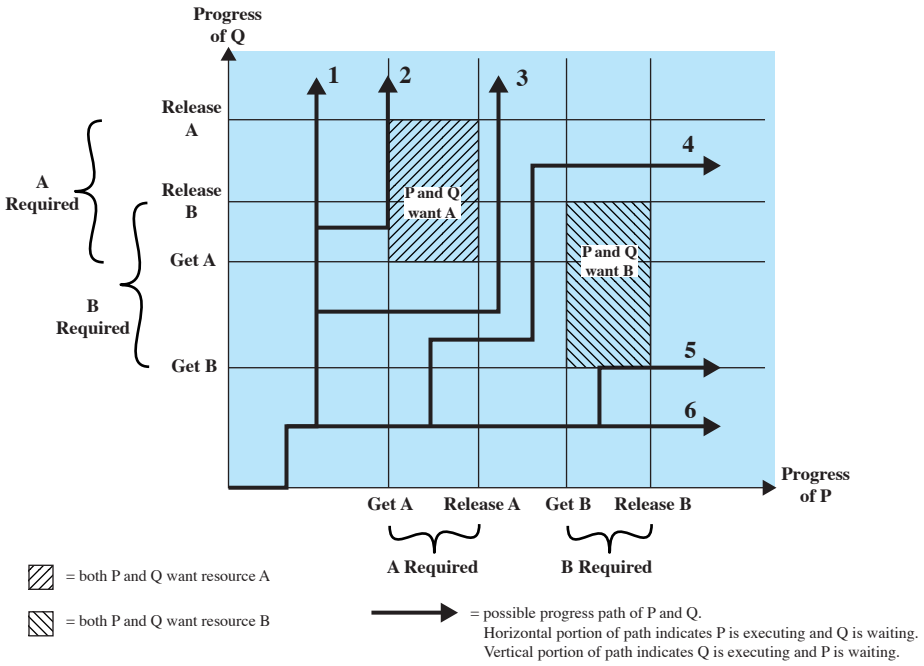

Figure 6.3 Example of No Deadlock [BACO03]

Process P

Step	Action
p_0	Request (D)
p_1	Lock (D)
p_2	Request (T)
p_3	Lock (T)
p_4	Perform function
p_5	Unlock (D)
p_6	Unlock (T)

Process Q

Step	Action
q_0	Request (T)
q_1	Lock (T)
q_2	Request (D)
q_3	Lock (D)
q_4	Perform function
q_5	Unlock (T)
q_6	Unlock (D)

Figure 6.4 Example of Two Processes Competing for Reusable Resources

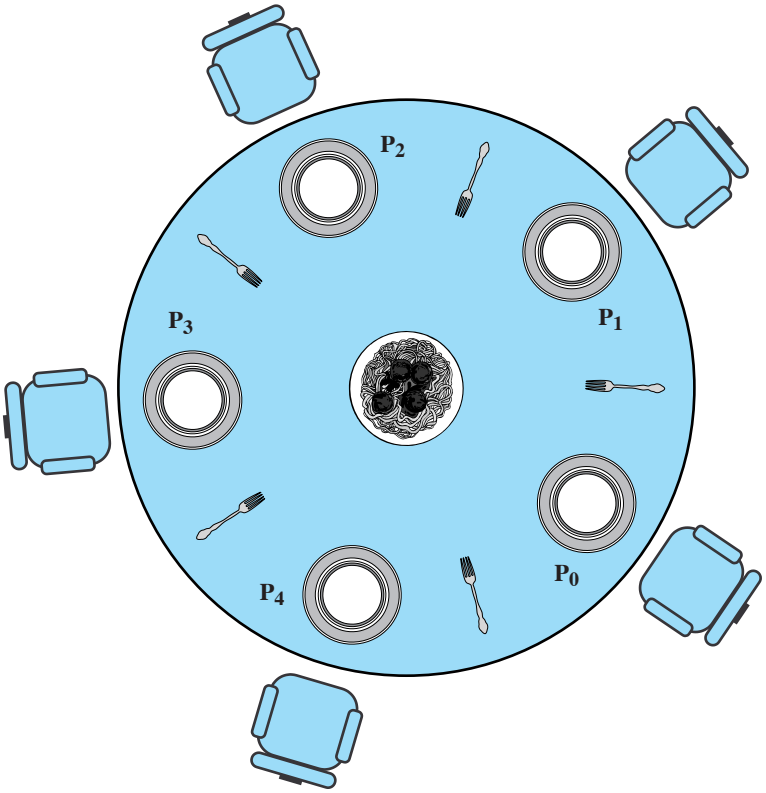

Figure 6.11 Dining Arrangement for Philosophers