Chapter 9: Distributed Mutual Exclusion Algorithms

Ajay Kshemkalyani and Mukesh Singhal

Distributed Computing: Principles, Algorithms, and Systems

Cambridge University Press

- Mutual exclusion: Concurrent access of processes to a shared resource or data is executed in mutually exclusive manner.
- Only one process is allowed to execute the critical section (CS) at any given time.
- In a distributed system, shared variables (semaphores) or a local kernel cannot be used to implement mutual exclusion.
- Message passing is the sole means for implementing distributed mutual exclusion.

- Distributed mutual exclusion algorithms must deal with unpredictable message delays and incomplete knowledge of the system state.
- Three basic approaches for distributed mutual exclusion:
  - Token based approach
  - Non-token based approach
  - Quorum based approach
- Token-based approach:
  - A unique token is shared among the sites.
  - A site is allowed to enter its CS if it possesses the token.
  - Mutual exclusion is ensured because the token is unique.

- Non-token based approach:
  - Two or more successive rounds of messages are exchanged among the sites to determine which site will enter the CS next.
- Quorum based approach:
  - Each site requests permission to execute the CS from a subset of sites (called a quorum).
  - Any two quorums contain a common site.
  - This common site is responsible to make sure that only one request executes the CS at any time.

Preliminaries

System Model

- The system consists of N sites, S_1, S_2, ..., S_N.
- We assume that a single process is running on each site. The process at site S_i is denoted by p_i.
- A site can be in one of the following three states: requesting the CS, executing the CS, or neither requesting nor executing the CS (i.e., idle).
- In the 'requesting the CS' state, the site is blocked and can not make further requests for the CS. In the 'idle' state, the site is executing outside the CS.
- In token-based algorithms, a site can also be in a state where a site holding the token is executing outside the CS (called the *idle token* state).
- At any instant, a site may have several pending requests for CS. A site queues up these requests and serves them one at a time.

Requirements

Requirements of Mutual Exclusion Algorithms

- **Safety Property:** At any instant, only one process can execute the critical section.
- **Liveness Property:** This property states the absence of deadlock and starvation. Two or more sites should not endlessly wait for messages which will never arrive.
- **Fairness:** Each process gets a fair chance to execute the CS. Fairness property generally means the CS execution requests are executed in the order of their arrival (time is determined by a logical clock) in the system.

Performance Metrics

The performance is generally measured by the following four metrics:

- **Message complexity:** The number of messages required per CS execution by a site.
- **Synchronization delay:** After a site leaves the CS, it is the time required and before the next site enters the CS (see Figure 1).

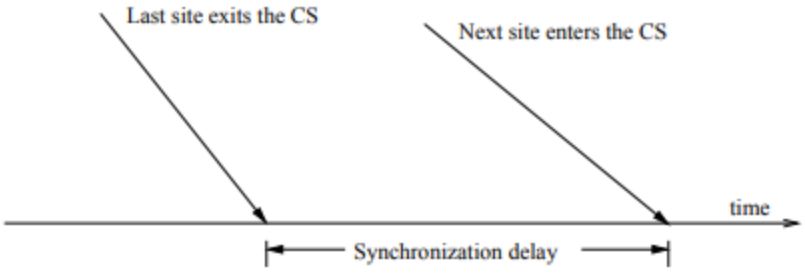

Figure 1: Synchronization Delay.

Performance Metrics

- **Response time:** The time interval a request waits for its CS execution to be over after its request messages have been sent out (see Figure 2).

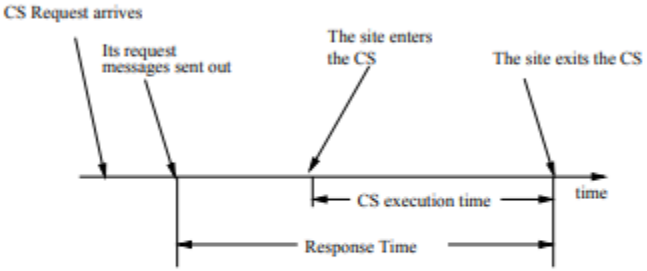

Figure 2: Response Time.

- **System throughput:** The rate at which the system executes requests for the CS.

$$\text{system throughput} = 1/(SD+E)$$

where SD is the synchronization delay and E is the average critical section execution time.

Performance Metrics

Low and High Load Performance:

- We often study the performance of mutual exclusion algorithms under two special loading conditions, viz., "low load" and "high load".
- The load is determined by the arrival rate of CS execution requests.
- Under *low load* conditions, there is seldom more than one request for the critical section present in the system simultaneously.
- Under *heavy load* conditions, there is always a pending request for critical section at a site.