Queuing system

queuing system : In the context of modeling and simulation, a queuing system is a mathematical model used to analyze the behavior of entities waiting in line to be served by a service facility. It is commonly used to study scenarios where entities, such as customers or tasks, arrive at a service point, wait in a queue if the service facility is busy, and then get served according to certain rules.

The key components of a queuing system include:

Arrival Process: Describes how entities arrive at the system, usually following a specific probability distribution.

Service Process: Defines how long it takes to serve an entity, also typically following a probability distribution.

Queue Discipline: Specifies the rules used to determine which entity gets served next when multiple entities are waiting in line.

Number of Servers: Determines the number of service facilities available to serve the entities simultaneously.

By analyzing a queuing system, researchers can gain insights into the system's performance, such as average waiting times, utilization rates of the service facility, and the probability of the system being in certain states. This analysis is valuable for optimizing system efficiency, identifying potential bottlenecks, and making informed decisions to improve the overall performance of real-world systems such as call centers, computer networks, and transportation systems.

Example:

Let's consider a simple example of a coffee shop to explain the queuing system.

In this coffee shop, customers arrive to order their coffee and wait in a single line (queue) to be served by the barista. The coffee shop has only one barista to serve the customers.

Arrival Process: Customers arrive at the coffee shop randomly, following a Poisson distribution. This means the time between successive customer arrivals follows a specific statistical pattern.

Service Process: The time taken by the barista to serve each customer follows an exponential distribution, representing the average time it takes to prepare a coffee.

Queue Discipline: The coffee shop follows a First-Come-First-Serve (FCFS) discipline, where the customer who arrives first in the queue is served first.

Number of Servers: There is only one barista available to serve the customers.

With this queuing system, we can analyze various performance metrics, such as the average waiting time for customers, the average time a customer spends in the coffee shop, and the utilization rate of the barista. By studying these metrics, the coffee shop owner can optimize staffing levels, estimate how many customers can be served in a given time, and improve the overall customer experience.

different queuing disciplines

In queuing theory, different queuing disciplines refer to the rules and policies used to determine the order in which entities (customers, tasks, or jobs) are served from the queue. The choice of queuing discipline can significantly impact the performance and behavior of the queuing system. Here are some common queuing disciplines:

First-Come-First-Serve (FCFS): In this discipline, the entity that arrives first in the queue is served first. It follows a strict order based on arrival time, ensuring fairness but not necessarily optimal in terms of minimizing waiting times.

Last-Come-First-Serve (LCFS) or Stack: In this discipline, the most recent arrival is served first, causing newer entities to jump ahead of older ones in the queue. This is commonly used in certain real-time systems or stack-based data structures.

Priority Queuing: Entities are assigned different priority levels, and those with higher priorities are served before those with lower priorities. This allows for the prioritization of specific types of entities but can lead to potential starvation of lower-priority entities.

Round-Robin (RR): In this discipline, each entity is given a fixed time slice or quantum of service, and the entities are served in a cyclic manner. If an entity's service time exceeds the quantum, it is moved to the back of the queue.

Shortest Job Next (SJN) or Shortest Job First (SJF): The entity with the shortest service time is served first. This aims to minimize average waiting times and is effective when service times are known in advance.

Priority-Preemptive Queuing: Similar to priority queuing, but in this case, higher-priority entities can preempt the service of lower-priority entities, even if they are being served.

Processor Sharing: In this discipline, the available resources are shared equally among all entities in the queue. Each entity receives a fraction of the resource, and all entities are served simultaneously.

Randomized Queuing: The order of service is determined randomly, providing equal chances for all entities to be served next. This can be useful when fairness is more critical than optimization.

The choice of queuing discipline depends on the specific requirements and goals of the queuing system. Different disciplines have their advantages and limitations, and selecting the most appropriate one involves considering factors like fairness, response time, resource utilization, and system efficiency.

examples of First-Come-First-Serve (FCFS) and Last-Come-First-Serve (LCFS) queuing disciplines:

First-Come-First-Serve (FCFS) Example:

Imagine a single-server ticket counter at a movie theater. As customers arrive at the counter to purchase tickets, they join a single queue. The ticket seller serves customers in the order they arrived at the counter. The first customer in line is served first, then the second, and so on. FCFS ensures that customers are served based on their arrival time, providing a fair and straightforward service order.

Last-Come-First-Serve (LCFS) Example:

Consider a real-time data processing system where data packets are being received and processed. As data packets arrive at the processing unit, they are immediately placed at the front of the queue. The processing unit then picks up the last-arrived packet and processes it first. This means newer packets get priority over older ones. LCFS is commonly used in certain real-time systems, where the most recent data is more critical or time-sensitive than the older data.

Kendall notation:, named after the British statistician David G. Kendall, is a standard notation used to describe and represent queuing systems in queuing theory. It provides a concise way to represent the key characteristics of a queuing model.

The Kendall notation consists of three parts, represented as A/B/C:

A: Arrival Process

Describes the distribution of inter-arrival times (time between consecutive arrivals).

Common values for A:

M: Exponential distribution (Poisson arrival process)

D: Deterministic arrival process (constant inter-arrival times)

G: General distribution (when the exact distribution is specified)

B: Service Time Distribution

Represents the distribution of service times for entities at the service facility.

Common values for B:

M: Exponential distribution (Poisson service time)

D: Deterministic service time (constant service time)

G: General distribution (when the exact distribution is specified)

C: Number of Servers

Indicates the number of servers available to serve the entities in the system.

Common values for C:

1: Single server (for a single-channel system)

2: Multiple servers (for a multi-channel system)

Using the Kendall notation, different queuing models can be represented concisely. For example:

M/M/1: Represents a single-server queuing system with Poisson arrivals, exponential service times, and a single channel.

M/D/3: Represents a three-server queuing system with Poisson arrivals, deterministic service times, and multiple channels.

G/G/2: Represents a two-server queuing system with general distributions for arrivals and service times, and multiple channels.

The Kendall notation is valuable in analyzing and comparing queuing models, as it provides a standard way to convey the essential characteristics of the system without requiring lengthy descriptions. It helps researchers and practitioners understand queuing systems and optimize their performance in various real-world applications.

NOTE: FROM CHAPTER 9 READ 9.2, 9.2.2 AND 9.2.3