CS240 Operating Systems, Communications and Concurrency

Scheduling on Multiprocessor Systems

Performance Issues with Multiprocessing

Queuing Theory

Queue Organisation for Multiprocessor Systems

Other Scheduling Environments

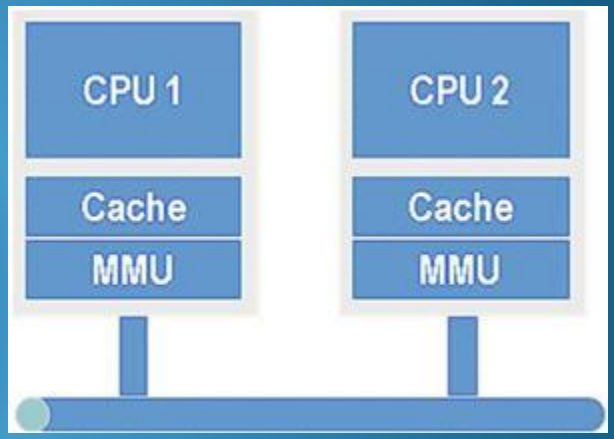
Distributed Systems

Real Time Systems

CS240 Operating Systems, Communications and Concurrency

Performance Issues with Multiprocessing

Using more processors seems an obvious way to increase overall system throughput but doubling processors does not usually double performance.



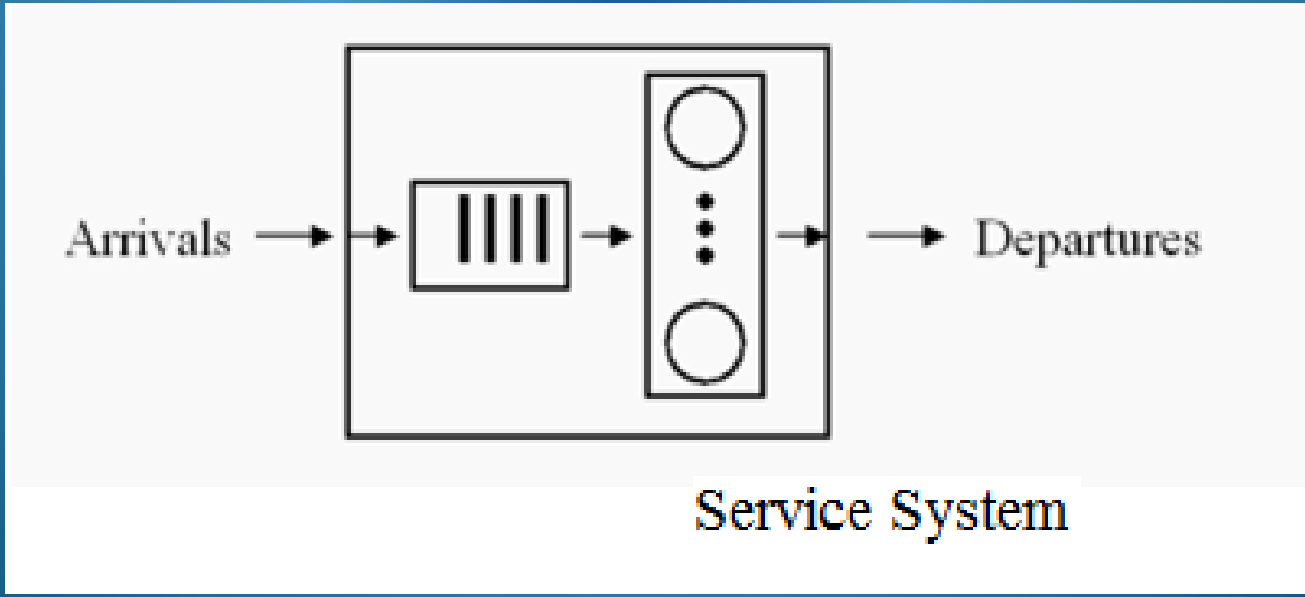
The scheduling problem is more complex and again there is no single best solution optimising processor utilisation for all cases.

Contention for the bus, I/O, shared memory, operating system code and maintaining cache coherency can cause some performance lag.

CS240 Operating Systems, Communications and Concurrency

Queuing Theory- The study of waiting lines or queues

Maximising throughput and reducing queuing times is a common aim of many real world services, not just computer systems and solutions require the application of queuing theory.



CS240 Operating Systems, Communications and Concurrency

Queuing Theory

In modelling and managing the performance of any system, we must address:-

The arrival pattern of tasks and their nature

How tasks are queued and dispatched

Processing capacity of the service

Every real world system might have its own unique arrival characteristics, physical queue configurations and service capacity and characteristic dynamics.

Queuing theory, modelling and simulation has the purpose of assisting managers of those services in how best to organise the system. You can look at the performance of a model before you construct a system.

CS240 Operating Systems, Communications and Concurrency

Arrival Pattern

In using real world services customers might arrive by

- Appointment to a consultant
- In groups to a restaurant
- At a particular time of night to a taxi service
- Randomly

Queuing and Dispatch Policy

There might be multiple queues, a VIP or FASTPASS priority queue or public/private queue or other type classification (heterogeneity) for example.

There may be a policy of customers being turned away if the service decides it is too busy or has no space to accommodate the queue and so on.

CS240 Operating Systems, Communications and Concurrency

Modelling the Arrival and Queuing Process

In a computer system, let's say the arrival pattern of tasks may be **random**, with single **independent tasks** arriving **one at a time**.

Many tasks might arrive close together sometimes, few at other times and maybe sometimes periods where none at all arrive but assume the arrival of tasks is completely random.

For modelling purposes we let λ represent the average rate of tasks arriving into such a system during an observation period.

What are the chances of more tasks in that period? We might like to design system to cope with fluctuations.

CS240 Operating Systems, Communications and Concurrency

Modelling the Arrival and Queuing Process

One way to estimate the arrival distribution might be to monitor the system and note the time between task arrivals and then using statistical techniques, fit a probability formula to the data so that we can estimate the likelihood of particular numbers of arrivals in any given period.

A **Poisson Distribution**, for example, can be used with the arrival process we described to our computer system in order to find the probability of a specific number of tasks arriving within a given time period.

CS240 Operating Systems, Communications and Concurrency

To fit this probability formula, arrival pattern of tasks with rate **λ** must be **random**, with single **independent tasks** arriving **one at a time**.

Probability of events for a Poisson distribution

An event can occur 0, 1, 2, ... times in an interval.

The average number of events in an interval is designated λ (lambda).

The probability of observing k events in an interval is given by the equation

$$P(k \text{ events in interval}) = \frac{\lambda^k e^{-\lambda}}{k!}$$

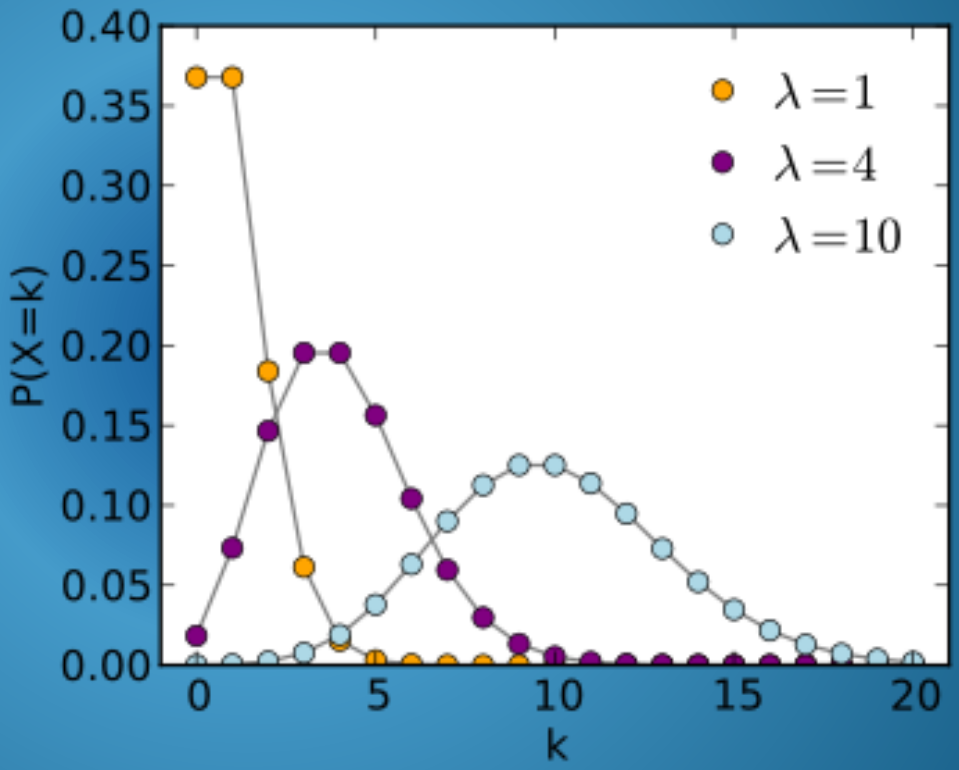
where

- λ is the average number of events per interval
- e is the number 2.71828... (Euler's number) the base of the natural logarithms
- k takes values 0, 1, 2, ...
- $k! = k \times (k-1) \times (k-2) \times \ldots \times 2 \times 1$ is the factorial of k.

This equation is the probability mass function (PMF) for a Poisson distribution.

CS240 Operating Systems, Communications and Concurrency

Poisson Distribution



For different values of the average arrival rate (lambda), the vertical access gives the probability of k events occurring in the given interval over which the average λ applies.

CS240 Operating Systems, Communications and Concurrency

Queuing Process & Service System

If the service system is busy, the arriving tasks are placed in a queueing system waiting for service.

The queueing system may have a particular configuration and /or constraints such as being composed of a single queue or multiple queues with finite resources (space), and serviced according to a particular discipline,

Tasks are dispatched from the queue to the service system when resources are available based on queue service policy.

CS240 Operating Systems, Communications and Concurrency

Modelling the Arrival and Queuing Process

We assume all computing tasks can be accommodated in the queue and will remain in the system until completed.

In real world queuing systems, the arrival and queuing pattern might be more dynamic,

Balk at joining long queue

Join for a while and leave due to excessive waiting

Jockey between one queue and another

Here we ignore those effects on system performance and presume all our tasks arrive randomly, are patient and well behaved.

CS240 Operating Systems, Communications and Concurrency

The Service System

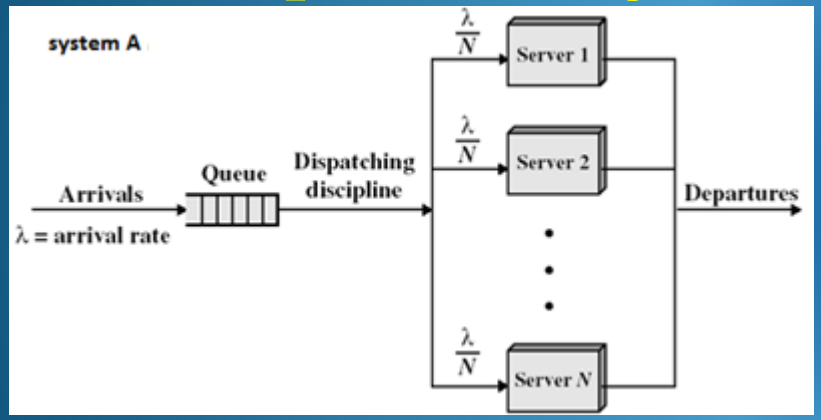
The service time may be fixed or may follow a distribution pattern – e.g. Exponential Probability Distribution

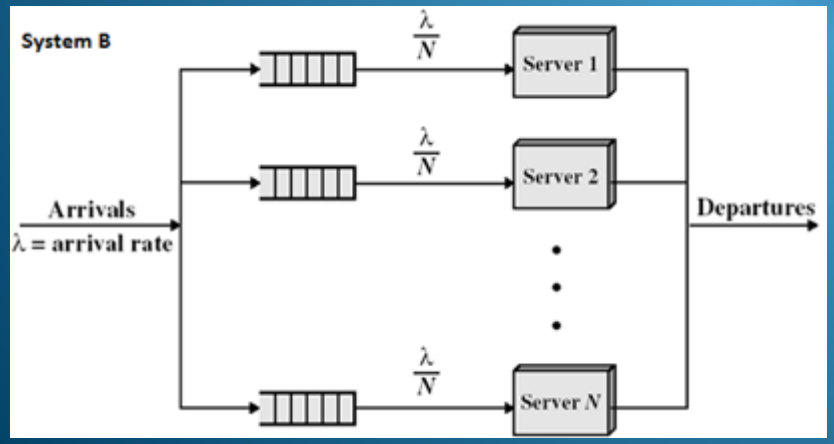
The service rate μ describes the number of tasks expected to be serviced during a particular time period, and the service time indicates the amount of time needed to service a task.

One is the reciprocal of the other. So if a processor can service 5 tasks in 1 second the service rate μ is 5 tasks per second and the service time is 0.2 seconds (1/5) per task on average.

CS240 Operating Systems, Communications and Concurrency

Multiprocessor System Organisation



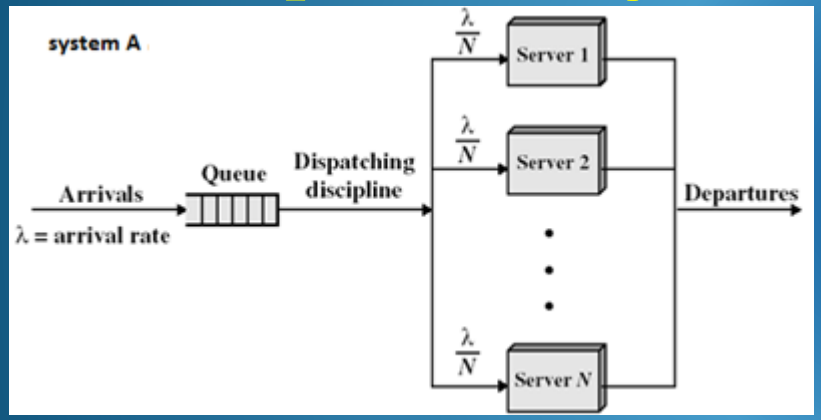


A multiprocessor system can be composed of a ***heterogeneous*** or ***homogenous*** collection of processors.

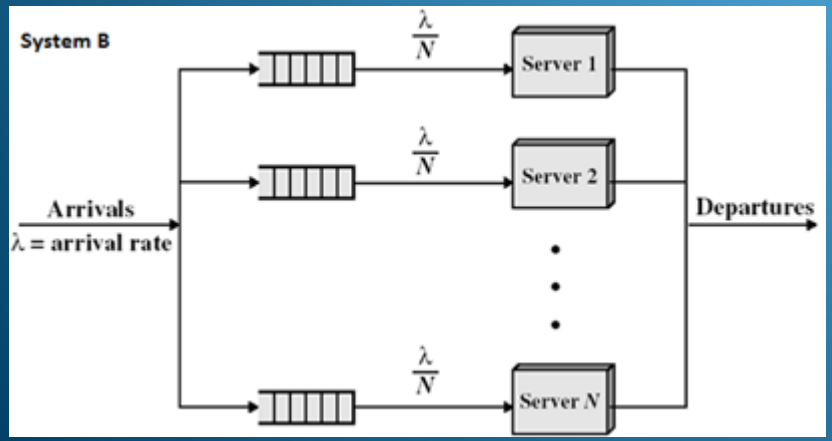
With homogeneous processors, a single queue of tasks may be serviced by all processors. A homogeneous system can use asymmetric or symmetric multiprocessing .

CS240 Operating Systems, Communications and Concurrency

Multiprocessor System Organisation



Serpentine queues are commonly used in banks, ticket counters where the tellers perform symmetric multiprocessing.

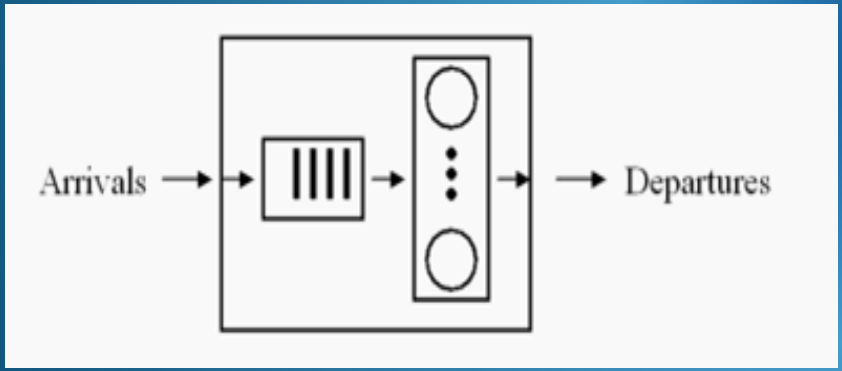


Multiple queues are used at supermarket checkouts due to physical constraints on queue space and travel time perhaps.

CS240 Operating Systems, Communications and Concurrency

Multiprocessor System Organisation

(Assuming Poisson Arrivals and Exponential Service Distribution)



If the average arrival rate of tasks to a system is λ and μ is the service rate then it can be proven (from Little's Law) that the mean residence or turnaround time (time spent in the box) *T*, is related to λ and μ by the formula:-

$$T = \frac{1}{\mu - \lambda}$$

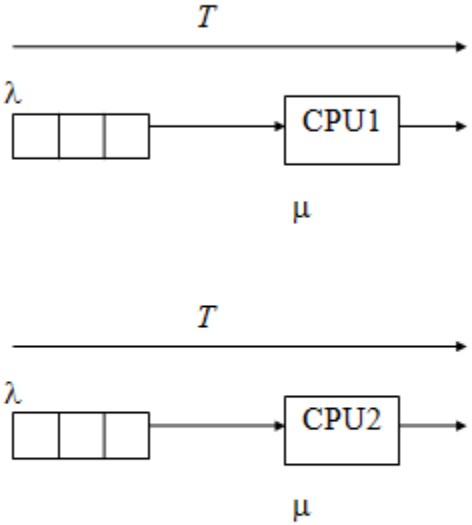
Of course μ must be greater than λ for the system to be able to cope with the load.

CS240 Operating Systems, Communications and Concurrency

Multiprocessor System Organisation

The Supermarket Model

If we take n processors, each with its own queue then we get:-



where

$$T = \frac{1}{\mu - \lambda}$$

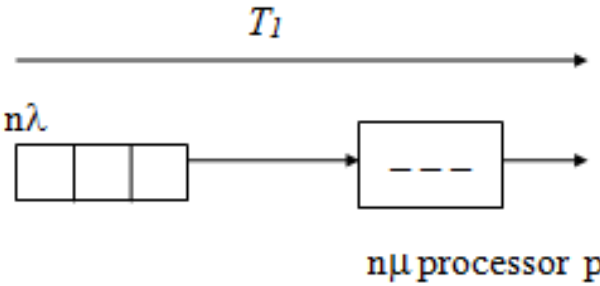
for each processor.

CS240 Operating Systems, Communications and Concurrency

Multiprocessor System Organisation

The Bank Model

Instead if we combine the queues into a single queue serviced by n processors we get:-



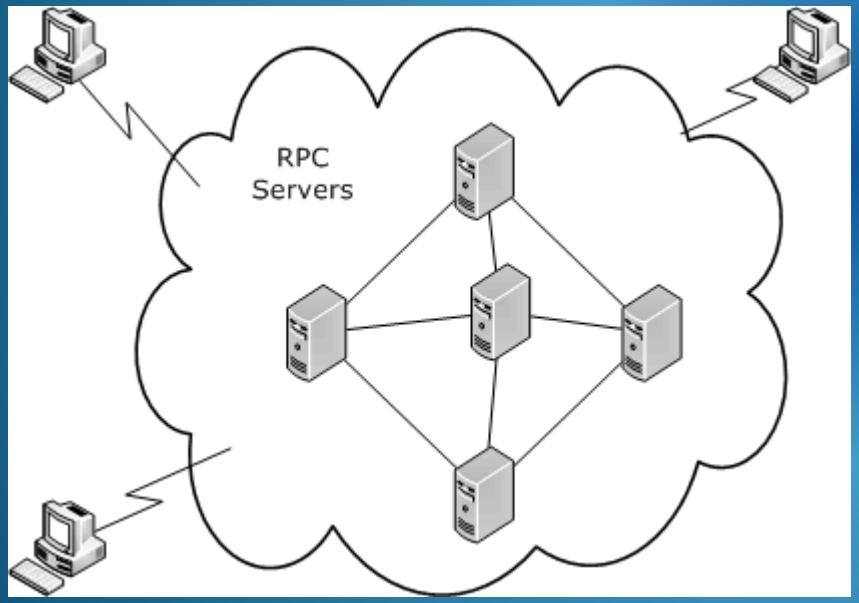
where

$$T_1 = \frac{1}{n\mu - n\lambda} = \frac{T}{n}$$ i.e a Faster Mean Turnaround Time

Assuming of course that all processors can be made equally busy all of the time and the arrival rate is random.

CS240 Operating Systems, Communications and Concurrency

Other Scheduling Environments



A **Distributed System** is composed of a collection of physically distributed computer systems connected by a local area network.

From a scheduling perspective, a distributed operating system would endeavour to monitor and **share** or **balance the load** on each processing node.

CS240 Operating Systems, Communications and Concurrency

Scheduling in Distributed Systems

Scheduling in distributed systems is even more complex because the system information is generally not accessible in one single place but is managed on separate nodes in the network.

Algorithms must gather information needed for decision making and transmit control information using message exchanges across the network. Message communication can be subject to delays due to corruption and loss, requiring occasional retransmission, or busy scheduling patterns, all leading to greater complexity within distributed algorithms.

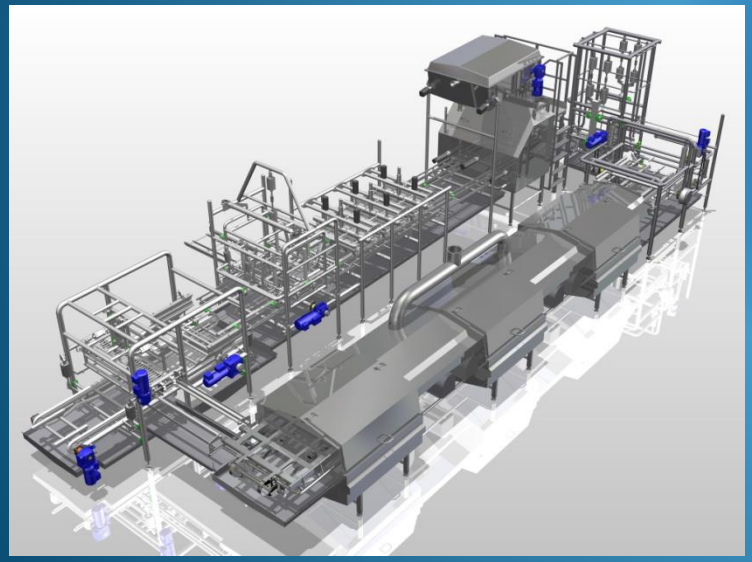
CS240 Operating Systems, Communications and Concurrency

Real Time Systems

Some computer systems are designed for specific industrial process control applications.

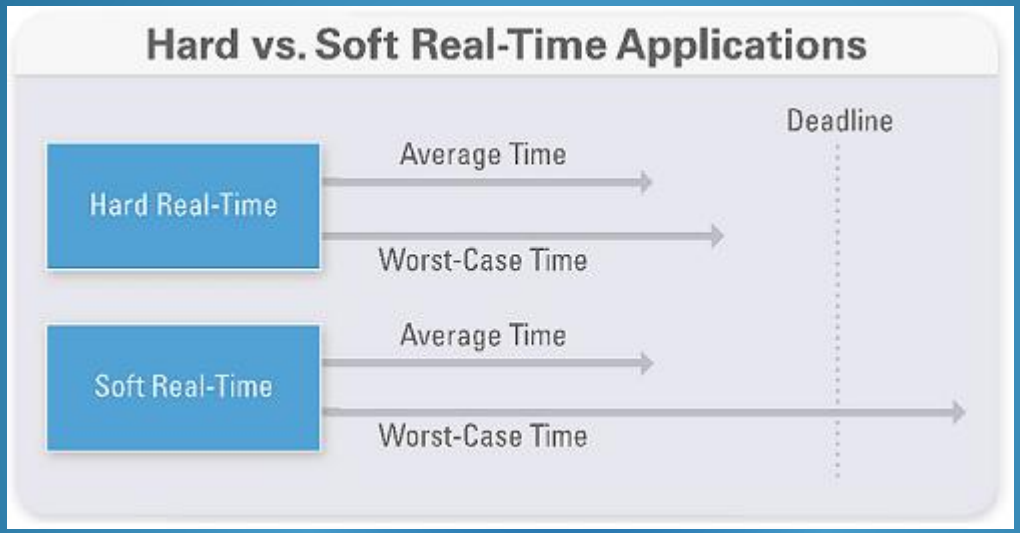
For example, consider a production line where raw materials are supplied at one end of the line and finished goods come out the other.

CS240 Operating Systems, Communications and Concurrency



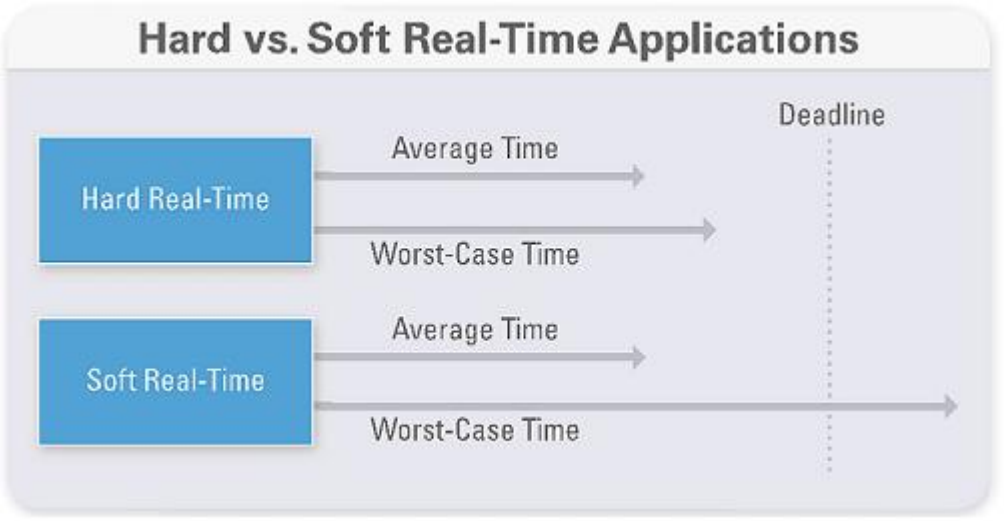
Along the production line, sensors, valves and other actuators are monitored and controlled by the computer system. Sensors bring data to the computer which must be analysed and subsequently results in modifications to valves and actuators. A real time system has well defined, fixed time constraints. Processing must be done within those constraints or the system may fail.

CS240 Operating Systems, Communications and Concurrency



Hard Real Time Systems are required to complete a critical task within a guaranteed amount of time. The execution time of code needs to be analysed before a task may be accepted by the scheduler.

CS240 Operating Systems, Communications and Concurrency

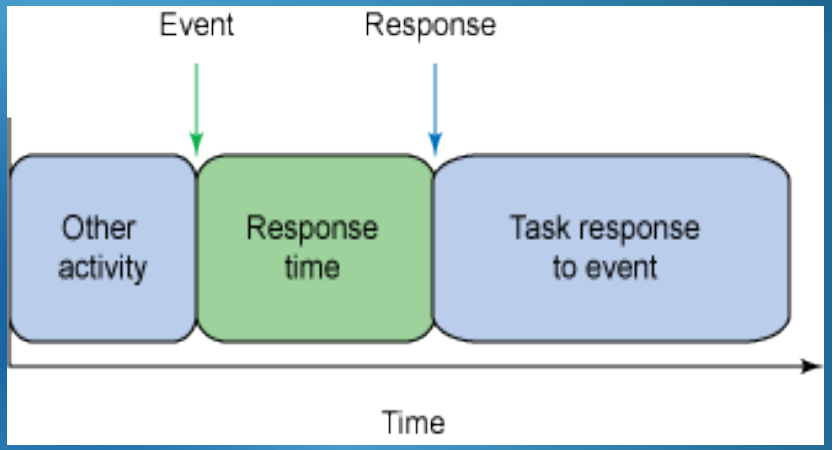


Soft Real Time Systems are ones which endeavour to meet scheduling deadlines but where missing an occasional deadline may be tolerable.

This can be achieved in general purpose systems by giving critical processes a **higher priority** than others, and their priority does not degrade over time.

CS240 Operating Systems, Communications and Concurrency

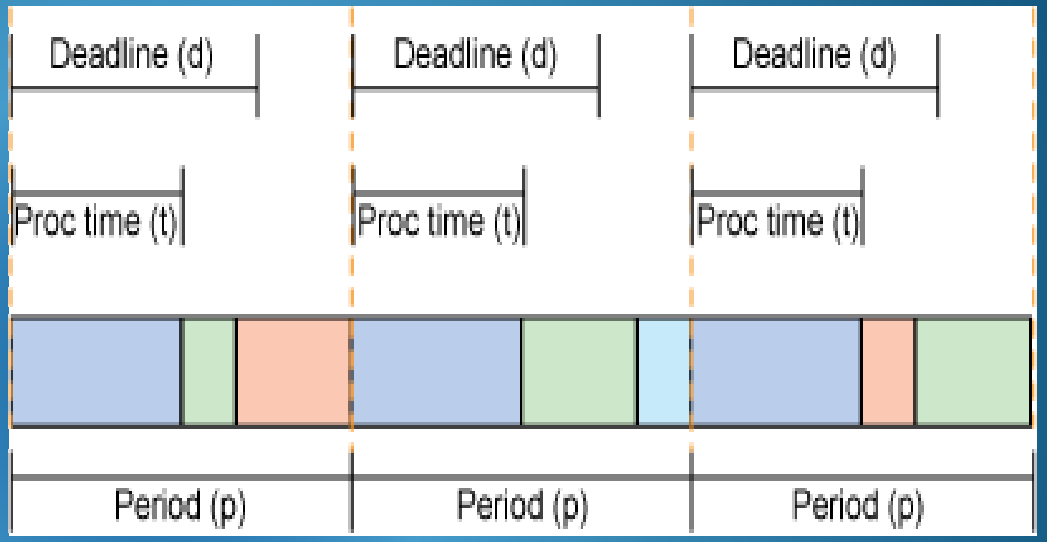
Real Time Systems



One of the key things when priority based approaches are used it that the **dispatch latency** must be small to ensure that a real time process can start executing quickly. This requires that system calls be preemptible. It must be possible to preempt the operating system itself if a higher priority real time process want to run.

CS240 Operating Systems, Communications and Concurrency

Dedicated Real Time Scheduling Systems

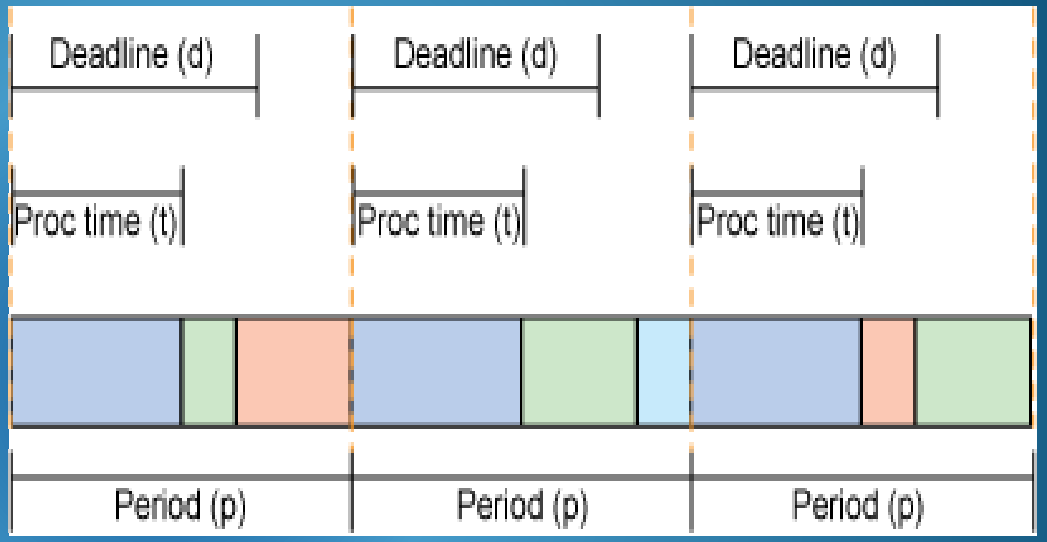


Algorithm Example - Earliest Deadline First

When an event is detected, the handling process is added to the ready queue. The list is kept sorted by deadline, which for a periodic event is the next time of occurrence of the event. The scheduler services processes from the front of the sorted queue serving the most urgent tasks first.

CS240 Operating Systems, Communications and Concurrency

Dedicated Real Time Scheduling Systems



Algorithm Example - Least Laxity Algorithm

If a process requires 200msec and must finish in 250msec, then its laxity is 50msec. The Least Laxity Algorithm chooses the process with the smallest amount of time to spare. This algorithm might work better in situations where events occurred aperiodically.