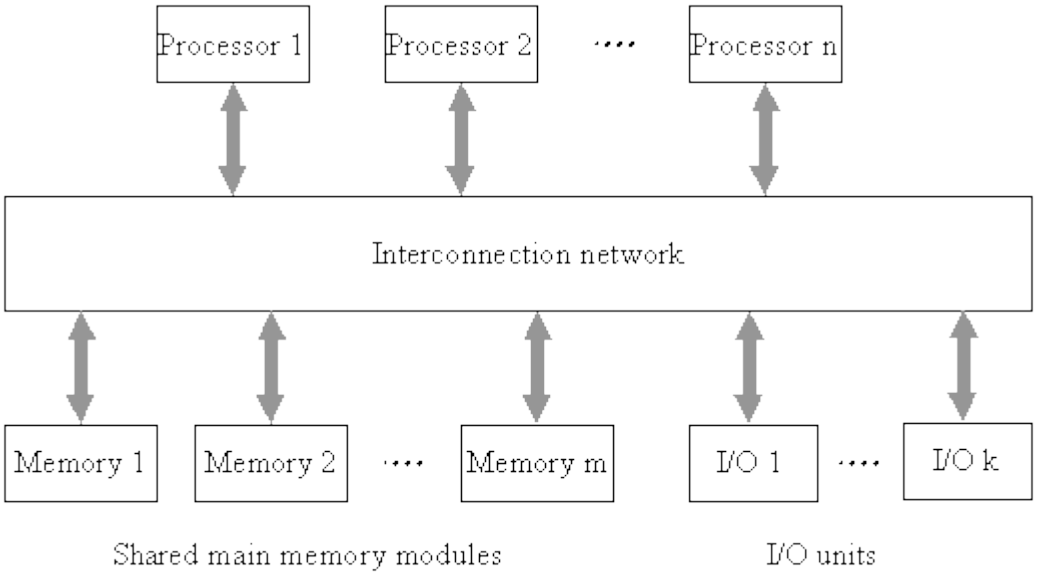
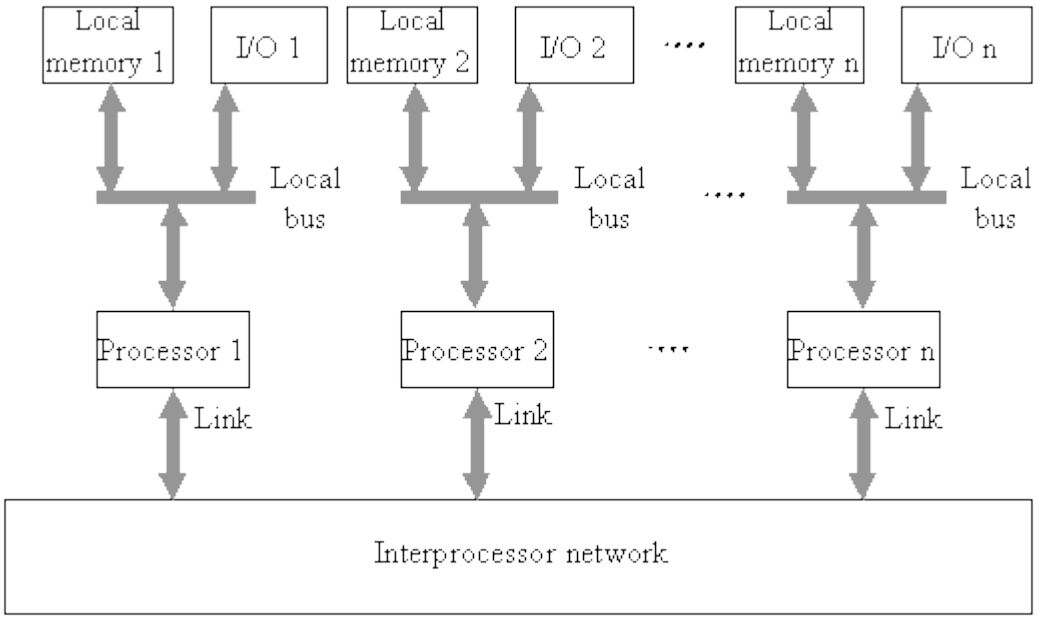
Chapter-9
Multiprocessor Scheduling Resource access control and Synchronization

Model of Multiprocessor and Distributed System

Multiprocessor system is tightly coupled system where global status and workflow information on all processor can be kept at a low cost. These systems may use centralized scheduler or each processor may have its own scheduler. Fig below shows a typical multiprocessor system.

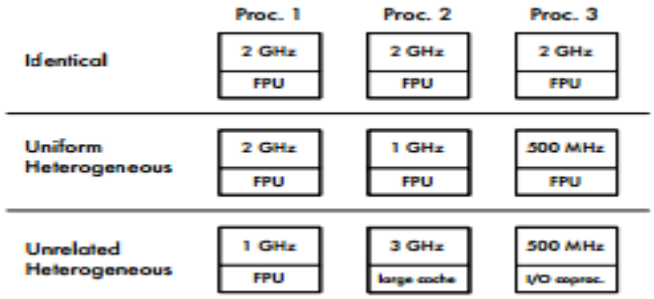


A distributed system is loosely coupled system where global status and workflow information update process is costly due to communication cost. Here each processor of the system has its own scheduler. Fig below shows the typical distributed system.



Identical vs. Heterogeneous Processors:

Identical processors can be used interchangeably such as each core of multi-core CPU. Heterogeneous processors cannot be used interchangeably such as disk drive, transmission line etc. fig below shows a typical processor organization of the identical as well as heterogeneous processors.



- If all processors of system have equal speed and capabilitie then they are called as identical processor.
- If all processors have equal capabilities but different speeds then they are called as uniform heterogeneous (or homogenous) multiprocessor.

If no regular relation assumed and tasks may not be able to execute on all processors then such system is called unrelated heterogenous multiprocessors.

Problems Associated with Resource Assignment in Multiprocessor Scheduling:

The major problrm in resource assignment in multiprocessor system is the task assignment problems. The task assignment problem is related with following two problems.

- The hard real time system are concern with the static system in which the task are prtitoned and statistically bound to the processor. The problem is how to partition the system of task and passive resources into modules and how to assign the modules to individuals processors.
- Another problem is related with the interprocessor communication and synchronization . A synchronization protocol is required to maintan precedence constraints of jobs on different processors.

Job shops and Flow shops:

According to job shop model, each task T_i in the system is a chain of n(i) jobs, denoted by $J_{i,k}$ for k = 1,2…..n(i). Adjacent jobs $J_{i,k}$ and $J_{i,k+1}$ on the chain execute on different processors. $J_{i,k+1}$ becomes ready for execution when $J_{i,k}$ completes its execution. Processor are specified for each job to run by the visit sequence $V_i = (v_{i,1}, v_{i,2}, \ldots\ldots v_{i,n(i)})$. Visit sequence $V_1 = (p_1, p_2)$ denote the job sequence has two jobs and first job execute on first processor and second job on second processor.

A flow shop is a special job shop which has identical visit sequence.

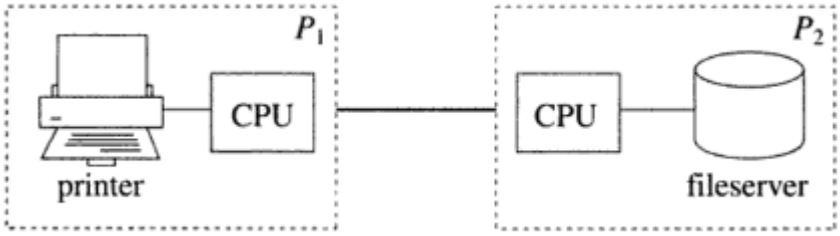
End to End Task:

Let the release time r_i of task T_i be the release time of first job $J_{i,1}$ in the task and the deadline d_i of the task is the deadline of its last job $J_{i,n(i)}$. As long as the last job completes by the task deadline, it is not important when other jobs of the task complete then such model is known as end to end task model.

Periodic End to End tasks:

An end to end task T_i is periodic with period p_i if the chain of n(i) jobs is released every p_i (or more) and the jobs in the chain execute in turn on processors according to visit sequence $V_i = (v_{i,1}, v_{i,2}, \ldots\ldots v_{i,n(i)})$.

Local versus Remote Resource(MPCP Resource Model):



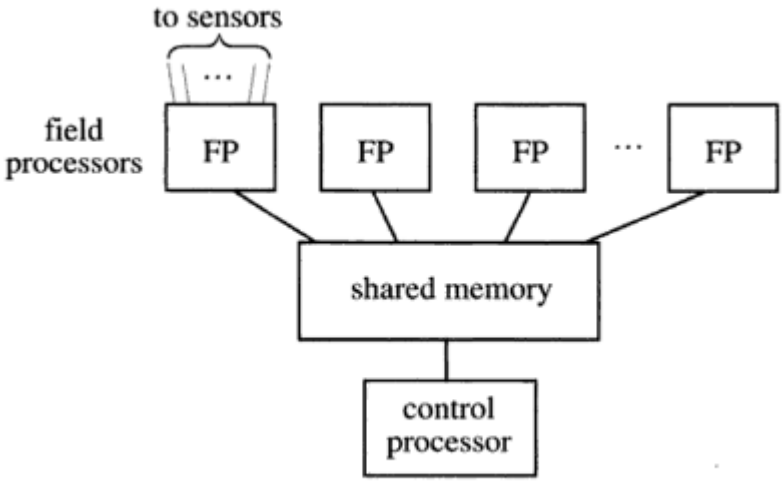
Here **P_1** is synchronization processor of resource printer while **P_2** is the synchronization of resource file server. Printer is local resource of J1 and J2 while fileserver is remote resource of J1. File server is global resource as it is needed by multiple jobs residing on different processor. When a job access global resource its global critical section executes on synchronization processor of that resource.

End to End resource model:

No jobs can make nested request for the resources residing on different processors.
J1 on P1 and P_1 → R1, P_2 → R_2
$L(R_1)$ $L(R_2)$ $U(R_2)$ $U(R_1)$ → not allowed
$L(R_1)$ $U(R_1)$ $L(R_2)$ $U(R_2)$ → allowed

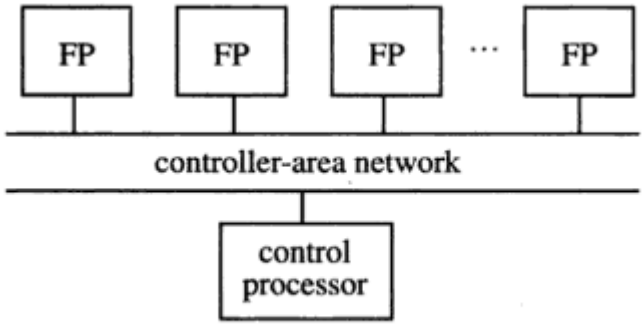
Inter-processor Communication:

Interprocess communication (IPC) is a set of programming interfaces that allow a programmer to coordinate activities among different program processes that can run concurrently in an operating system. This allows a program to handle many user requests at the same time. Since even a single user request may result in multiple processes running in the operating system on the user's behalf, the processes need to communicate with each other. The communication between two or more processor in a multi-processor organization to share the information/data is called inter-processor communication.
By using different algorithms, we can partitionton an application system into the compnents called modules and assigns the modules to processor. These algorithms make the use of information provided by interconnection parameters of jobs like volume of shared data stored on memory exchanged beteen each pair of jobs. The time reqire to synchronize jobs and transmit the data among them comes into account in several ways as inter-processor communication.



(a)

Fig(a). realtime monitor system that consists many field processor. Those jobs that collects and process the sensor data are called producer jobs since they generates the data. Those jobs that correlate and displays the data by executing on central processor are called consumer jobs. The jobs are communicated via. shared memory. Here each field processors are connected with shared memory via. a dedicated link that is shared by all field processors. The system contains three types of processors as field processor, shared-memory, control processor. The workload consists of end to end jobs, each job containing memory accessing jobs. In this way , delay in completion of each job caused by memory contention is taken into account. Hence shared memory cal be modeled as global resource whose synchronization processor is shared memory processor and both consumer and produceser requires this job.



(b)

Fig(b). shows a model in which all the producer are connected via. a network as controller area network(CAN). The result produced by each job on field processor are sent via network to control processor. Here network can be modeled as processor on which message transmission jobs are scheduled on a fixed priority basis. There is no need to account the interprocessor communication cost because they are taken into account by message transmission jobs in network.

Task Assignment:

In hard Real Time System, application system is partitioned into modules and each module is assigned to a processor, and most module assignment in done offline. Assignment is based on execution time requirement of job as the communication cost is minimal as well as minimization of communication cost . Communication cost of individual task is independent of where the task execute. In multiprocessor system task of same module may execute on different processors, the communication cost of these tasks is negligible due to shared memory architecture. During the early designed stage, we want to ignore it but it cost be depens on the system where task is executed.

Simple Bin Packing Formulation/ Algorithms:

- The task assignment problem can be generalized using a simple bin packing problem.
- There can be more than on job.
- The server utilization should not be greater be than 1 but if there are more than one job, the utilization can be grater than one. Therefore, all job can not execute on a single processor.
- The bin packing formulation is used to find optimal number of processor to execute all jobs to execute all jobs with total utilization less than 1.
- The binpacking algorithms can use two approaches
    a. Uniform size bin packing
    b. Variable size bin packing
- The variable size bin packing is the efficient method because it use the first fit method to select and execute the job on single processor.
- The job if single can be partitioned into different modules and by using bin packing formulation, an appropriate processor can be selected to execute each modules such that total utilization becomes less than 1.

Suppose a system with a 7 modules with utilization as 0.2 0.5 0.4 0.6 0.1 0.3 0.8 and bin size 0.9 then

Total utilization = 0.2 + 0.5 + 0.4 + 0.6 + 0.1 + 0.3 + 0.8 = 2.9

Different approaches like next fit, first fit and best fit are used to formulate the bin packing.

Next Fit

Algorithms:

- Keep a single open bin
- When an item does not fit, close this bin and open a new bin where Closed bins are never used again.
- In this method the bin packing be done as below.

Bin1	Bin2	Bin3	Bin4	Bin5
0.2	0.4	0.6	0.3	0.8
0.5		0.1		

First Fit

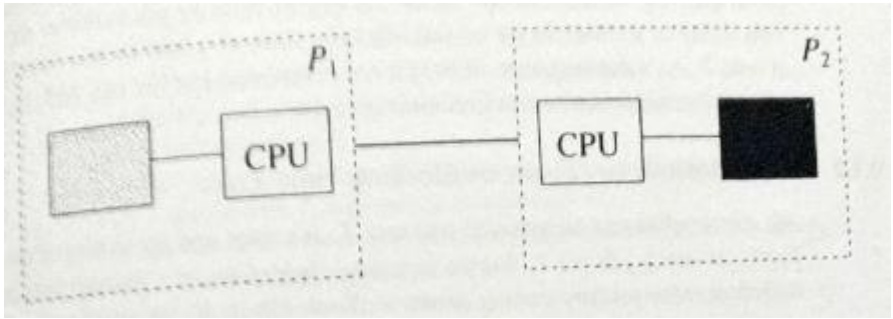
Algorithms:

- Keep all bins open .
- An item is assigned to the first bin in which it fits .
- check all previous bins to see if the next item will fit.
- If item does not fit in any open bin, open a new bin.

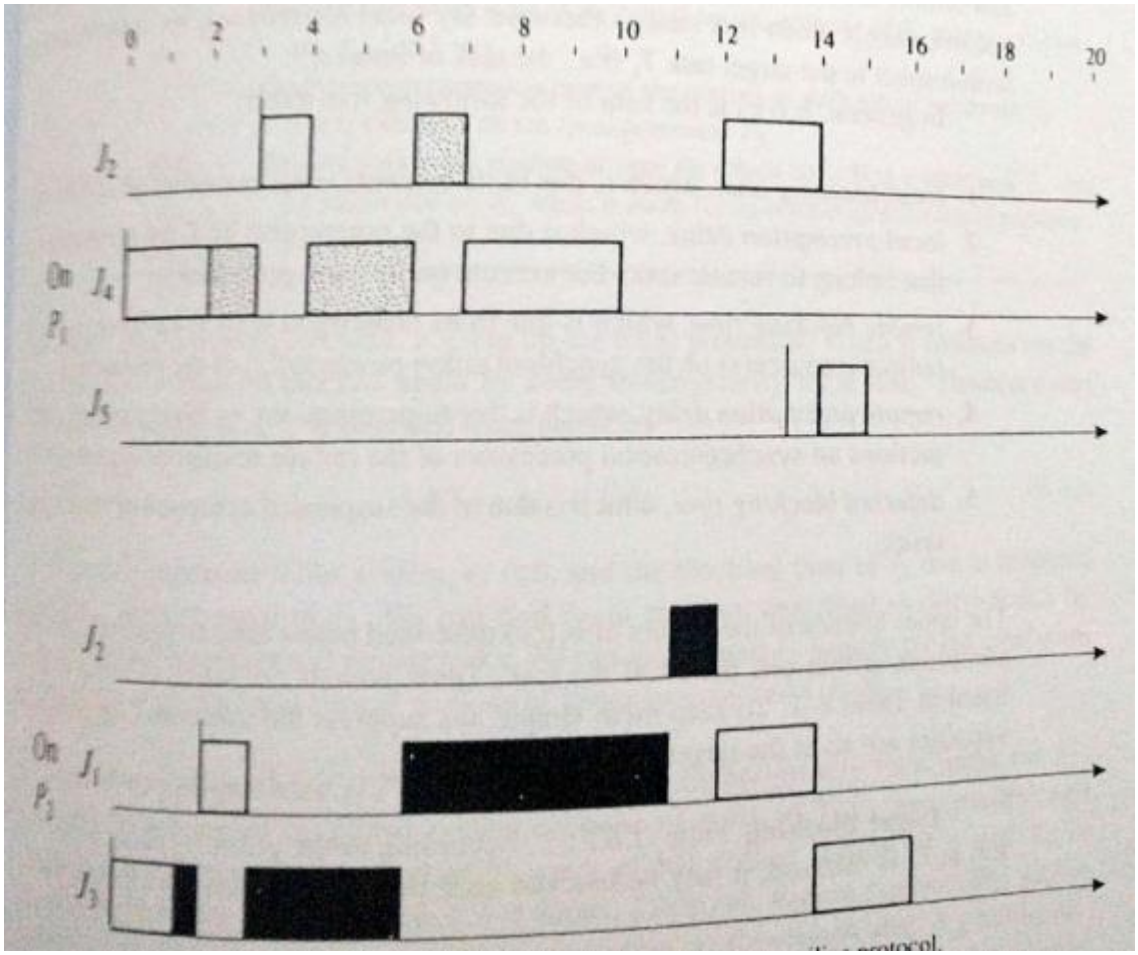
Bin1	Bin2	Bin3	Bin4
0.2	0.4	0.6	0.8
0.5	0.3		
0.1			

Multiprocessor Priority Ceiling Protocol:

MPCP assumes that taks and resources have been assigened and statically bound to process and scheduler of any synchronization processor know the priorities and resource requirement of all task requiring the global resources managed by the processor.
According to this protocol, the scheduler of each processor schedules all local task aswell as global critical section on a fixed priority basis. It also controls the resources excess according to priority ceiling protocol. When a task uses a global resource, its global critical section executes on its synchronization processor of resource. If global critical section task has lower priority than local task on synchronization processor then local task may delay the completion of global critical section and increases the blocking time of remote task. To avoid this, MPCP schedules all global critical section with higher priorities than local task. If the lower priority is π_{lower} then scheduler schedules the global critical section of a task with priorities π_i at priority $(\pi_i - \pi_{lower})$.
For example in a system of task with priorities 1 – 5 , global critical section of a task with priority 5 is scheduled at priority 0 which is higher than 1.
Consider a system with two processor P1 and P2 as shown in fig below.



The job J2, J4, J5 are local to processor P1 which is synchronization processor for resources dotted(R2). Dotted is the local resource because it is required by local jobs J2 and J4 only. Job J1 and J3 are local to processor P2 which is synchronization processor for the resource black (R1). Balck is the global resource required by J1, J2 and J3 as shown in timing diagram.



In the example, J2 is directly blocked by J4 when J2 requests dotted at time 4 and J1 is directly blocked by J3 at time 3 when it needs black. The execution of global critical section may be delayed by global critical section of other jobs on synchronization processor. For example, the global critical section of job J2 on P2 is delayed by global critical section of higher priority job J1 in time interval 6 to 11. This is called blocking time of J2.

At time 11, J1 exits from global critical section and its priority becomes lower than priority of global section of J2 i.e. priority of J1 is 1 and priority of J2 = 2- 5 = -3 which is higher than J1. As the result J2 preempts J1 on processor in the interval (11, 12]. The total delay suffered by a job due to preemption by global critical section of lower priority job is a considered as blocking time.

A job may be delayed by a local higher priority job whose execution is suspended on the local processor when higher priority jobs executes on a remote processor. This time is also considered as a blocking time in multiprocessor system. For example J2 is suspended at time 7 and it is still not completed in the time interval (13, 14) and J5 released at 13 can not start until time 14.

Elements of scheduling algorithm for end to end tasks:

Every periodic task that requires resource on more than one processor can be treated as an end to end periodic task. Each job of task contains a chain of component jobs which execute in

sequence on different processors. Suppose, task T_i has n(i) subtasks $T_{i,k}$ for k = 1, 2, …., n(i). These subtasks execute in turn in different processors to its visit sequence $V_i = (v_{i,1}, v_{i,2}, \ldots\ldots v_{i,n(i)})$ where $V_{i,k} = P_j$ which means the k^{th} subtask executes on processor P_j. The two essential components to the end to end scheduling schemes are 1) protocols for synchronizing the execution of sibling subtasks on different processor. So that precedence constraint among subtasks is maintained. 2) Algorithms for scheduling subtasks on each processor.

Inter-process Synchronization Protocol:

Greedy Synchronization Protocol

It is most commonly used in non-real time systems like in wide communication. Transmission of each newly compressed frame is made ready as soon as compression of frame is completed.

Greedy Synchronization protocol states that when j^{th} job of $T_{i,k}$ completes on $V_{i,k}$, the scheduler of $V_{i,k}$ sends a synchronization signal to the scheduler of $V_{i,k+1}$ on which the successor subtask $T_{i,k+1}$ executes. Upon receiving the synchronization signal, the scheduler of $V_{i,k+1}$ releases the corresponding job of task $T_{i,k+1}$.

Example: T_1 (6, 3) T_3 (6, 9, 4) $T_{2,1}$ (9, 3) and $T_{2,2}$ (9, 3)

These tasks run on two processors P_1 and P_2 where T_1 and $T_{2,1}$ run on P_1, $T_{2,2}$ and T_3 on P_2. T_1 has highest priority on P_1 and $T_{2,2}$ has highest priority in P_2.

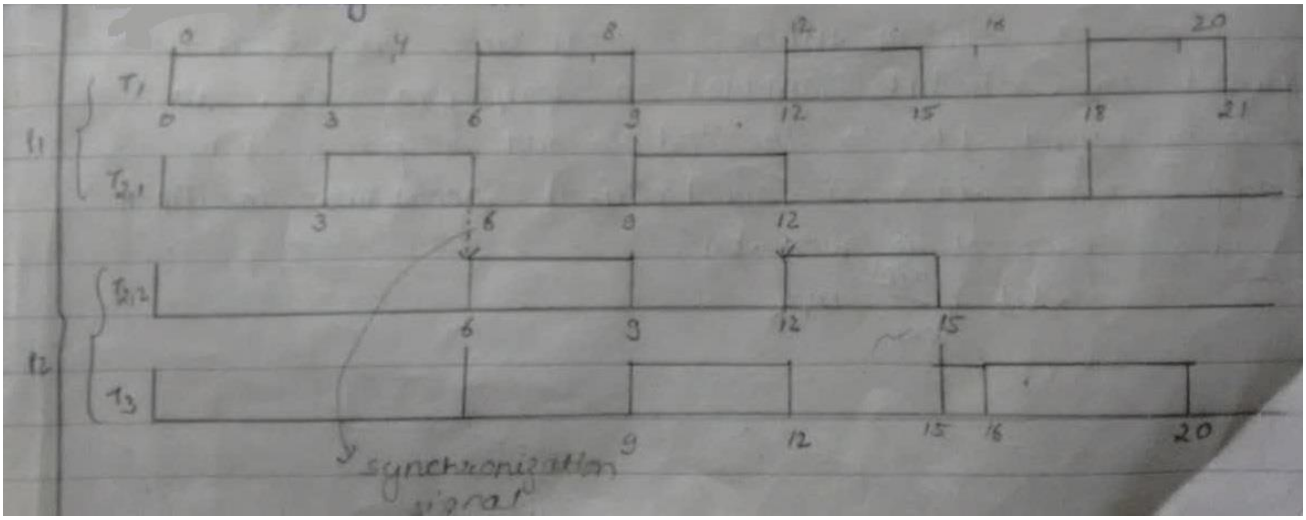


Figure: Illustration of Greedy Synchronization Protocol

Greedy synchronization protocol decreases the response time of end to end periodic task in comparison to non-greedy method. In the example above, the inter-release time of task T_2 is 6 while its period is 9 due to which T_3 misses its deadline.

End to End Task in Heterogeneous System

A simple embedded system contains different types of processors (CPU, disks, and networks). The scheduling algorithm used on each processor keeps the response time to a bound while satisfying the design constraints of the algorithm run by them. The bound is generated by applying non-greedy inter-processor synchronization. The modified phase modification protocol (MPM) and release guard protocol (RG) are such synchronization protocol. These protocols reshapes the release time of subtasks in different processors truly periodic.