*I.J. Modern Education and Computer Science*, **2014, 4, 31-37**
Published Online April 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2014.04.05

# A Correlation Framework for different Resource Access Protocol in Real Time System

**Mrs. Leena Das**
Assistant Professor KIIT University, Bhubaneswar, India
ldasfcs@kiit.ac.in

**Susmita Saha**
M.Tech Software Engg. KIIT University, Bhubaneswar, India
susmitauit@gmail.com

***Abstract***—We know various scheduling algorithm like EDF, RMA that are popularly used for sharing a set of serially reusable resources. But for share non preemptable resources those algorithm cannot satisfactorily be used. The main intent of resource access protocol is to schedule, and synchronizes the tasks when many tasks use same resources, means where resource is shared. So here in this paper the traditional techniques of resource access protocol which are used to share critical resources among a set of real time tasks, is discussed. This paper is mainly focusing on the different types of resource access protocols and their comparisons.

***Index Terms***—Real-Time System, Resource Access Protocol, PIP, PCP, HLP, MPCP, MSRP.

## I. Introduction

A system is called *Real time system*, when the system requires computable figures of time to express its performance. A real time system is a structure where the exactness of the system performance depends not only on the valid results of the computations, but also on the concrete time when the output is produced [9, 5]. Job or instance is software unit which is managed or executed by the operating system. The task is collection of related jobs collaborating to execute a function is called a task. Three types of task are there, first, *Periodic task*, where task repeats after a certain fixed time interval. Second, *Sporadic task*, recurs at random instants and third one is *Aperiodic task*, it is similar to the sporadic task. A whole system resource is divided into two types: Processors and Resources [2, 7]. Computers, Database servers are example of processor. Memory, Database locker are example of Resource. A job may need some resources through the processor during make progress. Each job $j_i$ is characterized by its parameters like, execution time, response time, release time, deadline [2]. *Execution time* is defined as how long the time it takes to run the given task. *Deadline* is defined as, time period within which the task must be done executing. The *Release time* is the instant when a job becomes available for execution. The *response time* is the length of time from the release time of the job to the time instant when it completes. A resource that can be used by more tasks is called *shared resource*. In this paper in section 2, why resource access protocol is used and their types are discussed. In section 3 the comparison between the resource access protocols is discussed. Section 4 concludes this paper and future work is discussed in section 5.

## II. Different types of resource access protocol

Task dependency occurs when result of one task is dependent on another task. Individual task is used these shared resource in exclusive mode. This means one task when it uses a resource, it cannot immediately hand over the resource to other task that requests the same resource which hold by that task at a specific time. The task handed over the resource only after completion its execution. These resources are called Non-preempted resources or critical section. A non-preemptable resource is also used in an exclusive mode. Therefore when a lower priority task holds the non-preemptable resource, a higher priority task will wait for this resource until the lower priority task completes using of resource. This situation is called Priority Inversion. And when higher priority task wait long time for resource, it may cause Deadlock. For this reason , The Scheduling algorithm as, EDF and RMA that are used for sharing a set of serially reusable resource like CPU, not used for shared non preemptable resource among a set of real time tasks. Scheduling of tasks that share critical resources lead two problems: Deadlock and Priority Inversion. Different protocols have been developed to avoid these problems [5, 7, 9].

There are different types of Resource Access protocol:

a. *Under Single processor system*

- Non-Preemptive Protocol (NPP)
- Highest Locker Priority (HLP)
- Priority Inheritance Protocol (PIP)
- Priority Ceiling Protocol (PCP)
- Stack Resource Policy (SRP)

b. *Under Multiprocessor system*

- Multiprocessor Priority Ceiling Protocol (MPCP)
- Multiprocessor Stack Resource Policy (MSRP)

*Sharing Resource:* Generally there are two classes of resource sharing, i.e., lock-based and lock-free synchronization protocols. In the lock-free approach [1,9], operations on simple software objects, e.g., stacks, linked lists, are performed by retry loops, i.e., operations are retried until the object is accessed successfully. The advantages of lock-free algorithm are that they do not require kernel support and as there is no need to lock, priority inversion does not occur. The disadvantage of these approaches is that it is not easy to apply them to hard real-time systems as the worst case number of retries is not easily predictable.

*Priority Inversion:* In Real-time system the priority based model stated that only higher priority task preempts the lower priority tasks. But the problem occurs when a higher priority process (H) requires resources (R) which is already hold by low priority task (L). The lower priority task does not handed over the resource to higher priority task until it has completed use of resource [8, 11]. In this case higher priority task has blocked by lower priority task.

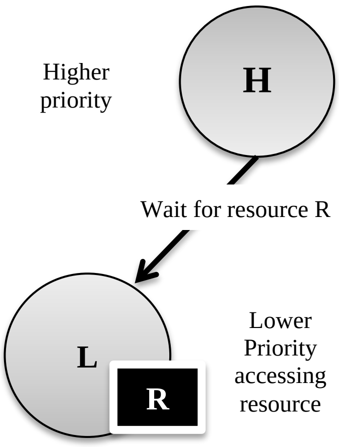

Fig. 1. Priority Inversion.

*Unbounded Priority Inversion:* It occurs when a higher priority task (H) needs a resource, which is hold by a lower priority task (L) then higher priority task waits for a lower priority task to release the resource. At that time if some intermediate (medium) priority tasks (I1, I2, I3,) are in ready to queue, which do not require R. In this case (H) would have to wait not only for L, but also for all intermediate priority tasks I1, I2, I3. In this case higher priority tasks (H) have to wait for the required resources for indefinite time in worst case. This situation is called unbounded priority inversion [8, 11].

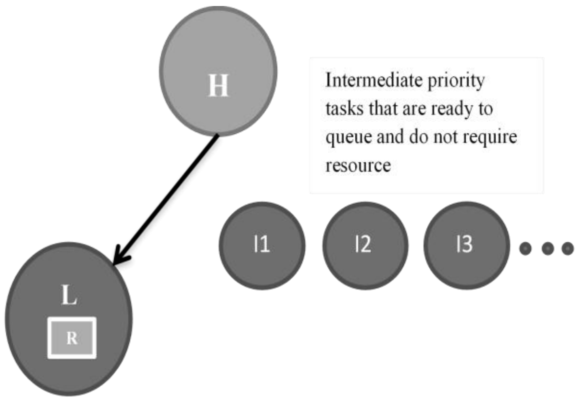
H
Intermediate priority tasks that are ready to queue and do not require resource
I1
I2
I3
L
R

Fig. 2. Unbounded priority ceiling protocol

*2.1. Non Preemptive protocol:* When a task enters in critical section, its priority is increased to maximum value. The simple equation of non-preemptive is represented as below:

$P_{CS} = \max \{p_1, \dots\dots, p_n\}$, where $P_{cs}$ is maximum priority value of all processor.

Table1. Advantage and disadvantage of NPP

| Advantage | Disadvantage |
|---|---|
| It is simple and easy to implement. | It causes Blocking because higher priority task which does not use shared resources. |
| Deadlock free. | |
| It is good resource access protocol when critical section is short. | |

*2.2 Priority Inheritance Protocol:* The priority Inheritance protocol by Sha et al [16] is based on two types of priority.

- *Assigned priority:* Priority which is assigned to job according to the scheduling algorithm [15].
- *Inherited priority:* Priority in which low priority job inherits priority of higher priority job. According to this protocol, all processes hold resource which are requested by lower priority process, inherits the priority of higher priority process until it uses the resource. After completion execution, the priority of lower priority process reverts to their original values.

In the Fig. 3, there are two jobs Ti, lower priority job and Tj, higher priority job. Ti the lower priority job with priority 5 holds the critical section, but when higher priority job Tj with priority 10 comes, and requests for critical resource CR, the lower priority job Ti inherits the priority of higher priority job Tj and the priority of Ti becomes 10. Higher priority job, Tj waits until lower priority job Ti finishes its execution. After completing execution Ti releases critical section and inherits its original priority i.e. 5 and Tj holds critical section CR [3].

Table2. Advantage and disadvantage of PIP

| Advantage | Disadvantage |
|---|---|
| PIP constrained priority inversion. | PIP cannot avoid chain blocking and deadlock. |
| It is very clear and simple to the user. | |

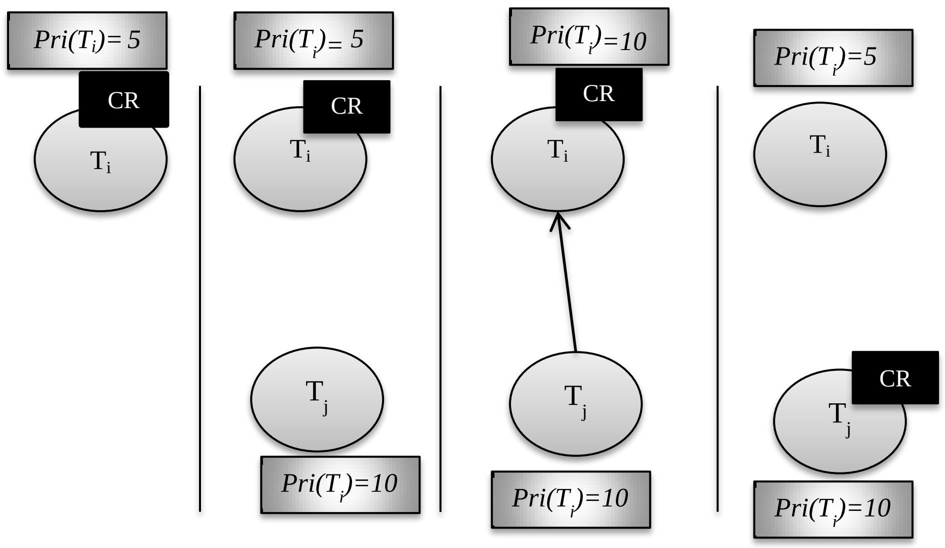

Fig. 3. Priority inheritance protocol

*2.3 Highest Locker Protocol:* Highest locker Protocol [HLP] is an expansion of Priority Inheritance Protocol [PIP]. HLP is mainly used to solve the problem of deadlock, chain blocking and also unbounded priority problem in PIP. HLP is a protocol in which, when a task is get into critical section, gets the highest priority among the task in the critical section. It means:

$$P_{cs} = \max\{P_k | \tau_k \; uses \; CS\}$$

Where $\tau_k$ tasks in are critical section and $P_k$ is the highest priority among all tasks in critical section. In HLP, *ceiling priority value* is assigned in every critical resource [3,9].Every critical resource is assigned a ceiling priority value; this ceiling priority value is the maximum value of the priorities of those tasks which send request to hold this critical resource. According to HLP, when a task holds a resource, the priority of task is determined to the ceiling priority of the resource.

If a task holds multiple resources, then it in inherits the highest ceiling priority of all its requested resources. As example,

Table3. Example1

| Task | Critical Resource | Priority |
|---|---|---|
| $T_1$ | $CR_1$ | 10 |
| $T_2$ | | 7 |
| $T_3$ | | 5 |
| $T_4$ | $CR_2$ | 9 |
| $T_5$ | | 8 |

From the above example, there are 5 tasks T1, T2, T3, T4, and T5. Tasks T1, T2, T3 request for critical resource CR1 and tasks T4 and T5 request for critical resource CR2. Now as the priority of critical section is following:

$$P_{cs} = \max\{P_k | \tau_k \; uses \; CS\}$$

So, the priority of critical resource $CR_1$ = max $\{T_1, T_2, T_3\}$ = max {10, 7, 5} = 10 and the priority of $CR_2$ = max$\{T_4, T_5\}$= max {9, 8} = 9. Therefore when either of $T_1, T_2,$ or $T_3$ acquires $CR_1$, its priority will be raised to 10. Another case is that if a task is holding more than one resource, its priority will become maximized of ceiling priorities of all resources it is holding. For example, if a task holds $CR_1$, and $CR_2$, would inherit maximum of those that is 10. In this way HLP solves the problem of deadlock in PIP.

Table4. Advantage and disadvantage of HLP

| Advantage | Disadvantage |
|---|---|
| HLP avoids Deadlock. | Opens up possibility for inheritance related inversion. |
| HLP avoids chain blocking. | |
| HLP also prevents unbounded priority problem. | |

*2.4 Priority Ceiling protocol:* The priority ceiling protocol first proposed by Sha, Lui, Rajkumar [16]. PCP is extended version of PIP and HLP, which is mainly used to resolve the problem of chain blocking, deadlock, and unbounded priority problem. It is also resolving the problem of HLP that is inheritance related inversion [14]. In PCP, every resources $CR_i$ associates with ceiling value$Ceil(CR_i)$. Current System Ceiling (CSC) tracks the maximum ceiling value of all resources. Therefore

$$CSC = \max\{ Ceil(CR_i) | \; where\; CR_i\; is\; currently\; in\; use\}.$$

As example below, four tasks $T_1, T_2, T_3, T_4$ share two non- preemptable resources$CR_1, CR_2$.

Table5. Example2

| Tasks | Critical Resources | Priority |
|---|---|---|
| $T_1$ | $CR_1 , CR_2$ | 10 |
| $T_2$ | $CR_1$ | 12 |
| $T_3$ | $CR_1$ | 15 |
| $T_4$ | $CR_2$ | 20 |

Now as per above example, $\mathrm{Ceil(CR_i)}$= max $(T_1, T_2, T_3)$ = 15 and Ceil $(CR_2)$ = max $(T_1 , T_4)$ = 20. Now $T_1$ is executing after acquiring $CR_1$ , then CSC is set to $\mathrm{Ceil(CR_i)}$= 15. There can be two situation occurs. Firstly, assume $T_4$ becomes ready. As $T_4$ has higher priority, it preempts $T_1$ and starts to execute. After some time $T_4$ requests to $CR_2$. As Priority ($T_4$) > CSC (15), $T_4$ granted the resource $CR_2$ and CSC is set to 20. When $T_4$ completes its execution, $T_1$ will get chance to execute [1,2]. Secondly, assume $T_3$ become ready, now as priority ($T_3$) >CSC (15), $T_3$ is not granted the $CR_1$.

Table6. Advantage and Disadvantage of HLP

| Advantage | Disadvantage |
|---|---|
| PCP protocol is free from deadlock and chain blocking | Suffers from context switches |
| Lower inheritance related inversion than HLP. | Opens up possibilities of high context switch overhead |

*2.5 Stack Resource Policy:* The SRP is similar to the PCP, proposed by T. Baker [6].When there are free resources then the task is allowed to execute. In this policy every resource is assigned a ceiling. The rule of SRP is that if pre-emption level > System ceiling, then higher priority job is allowed to start executing. A pre-emption level is assigned to each task in SRP. The relative deadlines of the tasks are reflected by preemption levels. Based on the maximum preemption level resources are given a ceiling values of the tasks that use resource at run time. When a task is released it can only acquire the currently executing task if its absolute deadline is shorter and its preemption level is higher than the highest ceiling currently locked resources. The result of this protocol is almost identical to PCP; tasks suffer only a single block, deadlocks are prevented, and a simple formula is available for calculating the blocking time [3, 4]. The SRP assures that once a job starts executing process, it cannot be blocked until completion; and only higher priority jobs can preempt this task [5].

In real-time system synchronization protocols are used to avert the unbounded priority inversion. In the background of multiprocessor synchronization, Multiprocessor Priority Ceiling Protocol [MPCP] extends PCP.

The MPCP is one and only synchronization protocol which worked in fixed priority scheduling in multiprocessor Policy [SRP] and access global resource.

Apart from these two real-time system, another synchronization protocol Multiprocessor Stack Resource Policy [MSRP], in which tasks synchronize local resources using Stack Resource synchronization protocols there is one protocol, which is Flexible Multiprocessor Locking Protocol [FMLP], which can be applied to both global and partitioned scheduling algorithms, that is Partitioned-EDF and Global-EDF. However we have discussed here only two protocols that are Multiprocessor Priority ceiling Protocol [MPCP] and Multiprocessor Stack Resource Policy [MSRP].

*2.6 Multiprocessor Priority Ceiling Protocol:* This protocol is extension of Priority Ceiling Protocol. According to the multiprocessor priority ceiling protocol, the scheduler of each processor schedules the entire *local tasks* and *global critical section*. When a task uses a global resource, its global critical section executes on the synchronization processor of the resource. If the global critical section of a remote task were to have a lower priority than some local tasks could delay the completion of global critical section and prolong the blocking time of remote task. To prevent this, the MPCP schedules all global critical sections at higher priorities than all local tasks on every synchronization processor. MPCP reduces the *remote blocking problem* [13]. Remote blocking occurs when a task has to wait for the execution of another task of any priority assigned to another processor [3, 6]. The MPCP is one and only protocol in multiprocessor which works under fixed priority scheduling. According to MPCP model a job may require both local and remote resources. The access of each resource is controlled by the scheduler of the synchronization processor of the resource. The critical section is when one process is executing in this section, then no other process is to be allowed to execute. In *global critical section (gcs)* tasks request for global resources and in local critical section tasks request for local resources [12].

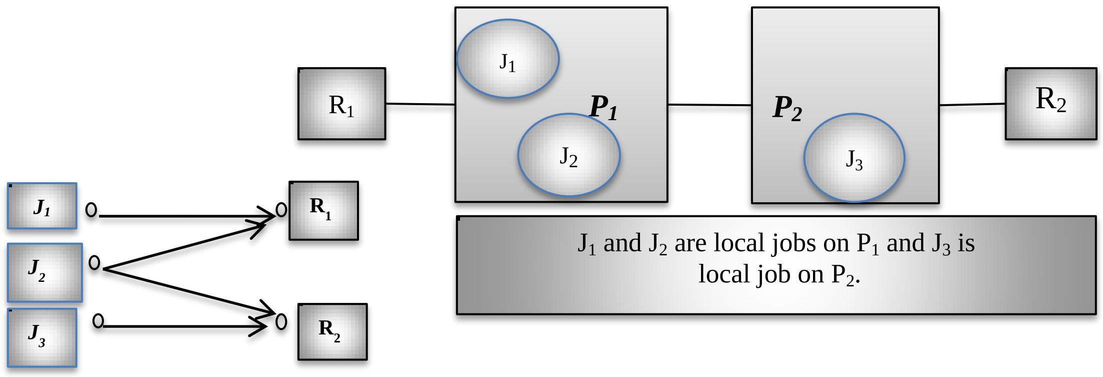
$J_1$
$J_2$
$J_3$
$P_1$
$P_2$
$R_1$
$R_2$
$J_1$ and $J_2$ are local jobs on $P_1$ and $J_3$ is local job on $P_2$.

Fig.4. Local and Global Resource model

As example, in fig4, there are two resources R1 and R2, two processors P1 and P2. Now Resource R1 resides on Processor P1, i.e. P1 is the synchronization processor of R1 and R2 resides on P2. Here are three jobs j1, j2, and j3, where j1 and j2 are local to P1 and j3 is local to P2.

- Now j1 executes on P1or it has only a local critical section.
- j3 requires only R2 ,since R2 resides on its local processor, this resource is local to j3.
- Suppose during the execution j2, it requires resource R2. J2 resides on P1 but it requires resource R2 which resides on processor P2 i.e., global critical section of j2 executes on P2. After completing global critical section of j2, it returns on P1 to execute.

So, In MPCP model, both local and global resources are required by a job. Scheduler of synchronization processors of each resource controls the access of each resource.

An approach for blocking time under MPCP is presented by Jane W.S. Liu in [3]. The author presents five blocking factor and the upper bounds to factor is the sum of these five blocking factors.

- Local Blocking Time [LBT]
- Local Preemption Delay [LPD]
- Remote Blocking Time [RBT]
- Remote Preemption Delay [RPD]
- Deferred Blocking time [DBT]

The upper bounds to factor bi(rc) is sum of the all these five factors.

- LBT occurs due to resource confliction the tasks are blocked on its local processor. According to priority ceiling protocol, after each job in Ti is released, it may be blocked once by a lower priority task. Moreover, each time when it uses a remote resource, it gives up the local processor. When it resumes on the local processor, it may be blocked again by some lower priority local task.
- LPD occurs due to preemption of tasks by global critical section which executes in its local processor but resides in remote task.
- RBT occurs due to confliction between lower priority tasks for acquiring remote resources. Each time when Task requests a global resource on a remote synchronization processor, its global critical section may be blocked once by a task that also requires some global resource on the processor and has a lower priority than task.
- RPD occurs when a task requires remote resource at that time higher–priority global critical section preempts that task.
- DBT occurs when execution of local higher priority task has been suspended.

The upper bound of the factors of bi(rc) is -equal to the sum of all these blocking factors as said Liu [3].

If we assume LBT, LPD, RBT, RPD and DBT as respectively b1, b2, b3, b4, b5 then bi(rc) = b1+b2+b3+b4+b5.

The MPCP is the only existing fixed priority scheduling in multiprocessor system.

*2.7 Multiple Stack Resource Protocol:* MSRP is proposed by Gai. et al [10, 11]. This protocol is extension of Stack Resource Protocol. MSRP is a partitioned-EDF based synchronization protocol in multiprocessor. According to MSRP, the task is allowed to use the local critical resources under the SRP policy. If a task tries to access a global critical resource that is already locked by a task on another processor, the task performs a busy wait, called a spinlock [6, 12, 13]. According to MSRP, *busy waiting (spin lock)* is defined as the processor is kept busy without doing any work. Busy waiting is performed by a blocked task on global resource. Thus the span of spin lock means, locking a global resource should be minimized. To minimize the duration of busy waiting the blocked tasks on a global resource are added into FIFO queue as F. Nemati says [4, 5]. So, MSRP is called spin-based synchronization protocol. A set of tasks sharing lock based resources which are classified as local or global resources. MSRP protocol quick fixes the problem of task allocation in resource sharing system. MSRP has similar property as MPCP such as,

- As PCP cannot directly apply to multiprocessor so MPCP is the extended version of MPCP, SRP cannot directly apply to Multiprocessor, for this reason extended version of SRP is MSRP.
- As in the MPCP algorithm the task suspended.
- Busy waiting or spin lock property is also here for MSRP.

## III. Comparisons between different protocols

Here in this section we have compared the overview, advantage and disadvantage of resource access protocols for uniprocessor and multiprocessors. Actually, the disadvantage of Priority Inheritance Protocol is eliminated in the Highest Locker Protocol and Priority Ceiling Protocol. PCP is also resolving the problem of HLP that is inheritance related inversion. From this comparison we can easily figure out which protocol will be suited for one system whether it has uniprocessor or multiprocessor.

### *3.1 Comparison between PIP, PCP, HLP:*

Table7. Tabular form of comparison between Resource accesses Protocol under single processor system

| **PIP** | |
|---|---|
| Overview: | All processes hold resource which are requested by higher priority process, inherits the priority of higher priority process until it uses the resource. After completion execution, the priority of lower priority process reverts to their original values. |
| Advantage: | Effectively overcomes the unbounded priority inheritance protocol. |
| Disadvantage: | May suffer from Chained blocking, and also does not prevent deadlock. |
| **HLP** | |
| Overview: | HLP is a protocol in which, when a task is get into critical section, gets the highest priority among the task in the critical section. |
| Advantage: | Avoids chain blocking, deadlock, unbounded priority problem |
| Disadvantage: | Opens up new problem Inheritance related inversion |
| **PCP** | |
| Overview: | In PCP, every resources $CR_i$ associates with ceiling value$Ceil(CR_i)$. Current System Ceiling (CSC) tracks the maximum ceiling value of all resources. |
| Advantage: | It removes deadlock and chained block. |
| Disadvantage: | Suffers from context switches and High context switch overhead and therefore cause to miss the deadlines. |

*3.2 Comparison between MPCP and MSRP:* When the time span of global critical sections (gcs) are increased then MPCP works better, but MSRP works better when critical section becomes shorter as discussed by Gai et al. [10, 11]. Author has compared between MPCP and MSRP with two case studies. Another point is that MSRP works better than MPCP where resources accessed by many resources or vice versa.

Table8. Tabular form of comparison between Resource accesses Protocol under Multiprocessor System

| **Subject** | **MSRP** | **MPCP** |
|---|---|---|
| Deadline miss ratio | More missed deadline caused more additional blocking. | Less missed deadline caused less blocking |
| Priority Scheduling ratio | MSRP's overhead is highest. | Overhead is lowest than MSRP. |
| Context switch ratio | Has lowest number of context switches | Context switches are higher than MSRP |

## IV. CONCLUSION AND FUTURE WORK :

This paper explores the co-relation among the resource access protocol for uniprocessor and multiprocessor real-time system. The resource access protocols for single processor are Priority Inheritance protocol, Higher Locker Priority, Priority Ceiling Protocol, and for multiprocessor are Multiprocessor Priority Ceiling Protocol and Multiprocessor Stack Resource Protocol. From the above discussion we can conclude that which resource access protocol can be more appropriate to schedule the tasks in different shared resource environment. Though it is a basic survey work as well as comparison among the different protocols, we can get a clear idea about the strength and weakness of different resource access protocols, which are easily applicable at what point of time.

Another point is MPCP is a new topic for research area. A new algorithm named Reduce Blocking under Multiprocessor Priority Ceiling Protocol [RBMPCP] can be introduced to reduce the blocking time by partitioning tasks into segments and put them together in different processors in the near future and can be proved our algorithm is better than the existing one.

## ACKNOWLEDGEMENT

We wish to thank who directly or indirectly supported our work. We thank our own affiliated organization KIIT University for providing us the right eco system to carry out our research work and all those who helped us to carry out our research. Due to which we are able to come out with this result. We thank our family members and one and all whose well wishes made us to reach our goal.

## REFERENCES

[1] Jian-Jun Han, Dakai Zhu, Xiaodong Wu, L.T.Yang, Hai Jin, “Multiprocessor Real-Time System with Shared Resources: Utilization Bound and Mapping”, IEEE Transaction on Parallel and Distributed System, Vol. XXX, No. XXX, Jan, 2013.

[2] Sara Afshar, Farhang Nemati, Thomas Nolte, “Towards Resource Sharing under Multiprocessor Semi-pPartitioned Scheduling,” *978-1-4678-2684-1/12, IEEE, 2012.*

[3] Jane. W. S. Liu “Real Time Systems” Pearson Education, thirteenth impression, 2012, P. 49-55.

[4] Farahang Nemati, “Resource Sharing in Real Time Multiprocessor”, Malardalen University, Sweden, 2012.

[5] Farhang Nemati, Moris Behnam and Thomas Nolte, “Imdependently-developed Real-Time Systems on Multi-cores with Shared Resources”, 23rd Euromicro Conference on Real-Time Systems, 1068-3070/11, IEEE, 2011.

[6] Andrew Carminati, Oliveira, “Intelligent priority ceiling protocol for scheduling”, LINDI 2011, 3rd IEEE International Symposium on Logistics and Industrial Informatics August 25–27, 2011, Budapest, Hungary.

[7] Apurva Shah, Ketan Kotecha, Adaptive Scheduling Algorithm for Real-Time Multiprocessor System, 2009 IEEE International Advance Computing Conference (IACC 2009), Patiala, India, 6-7 March, 2009.

[8] Jim Ras and Albert M. K. Cheng “An Evaluation of the Dynamic and Static Multiprocessor Priority Ceiling Protocol and the Multiprocessor Stack Resource Policy in an SMP System”, 15th IEEE Real-Time and Embedded Technology and Applications Symposium, 2007.

[9] Rajib Mall, “Real-Time Systems, Theory and Practice”, Dorling Kindersley (India) Pvt. Ltd., licensees of Pearson Education in South Asia, 2007.

[10] Paola Gai, Natale, Giuseppe Lipari, “A comparison of MPCP and MSRP when sharing resources in the Janus multiple-processor on a chip platform”, Proceedings of the 9th IEEE Real-Time and Embedded Technology and Application Symposium (RTAS’03), 2003, IEEE.

[11] P.Gai, G.Lipari, M.D. Natale, “Stack based minimization for embedded real time systems on a chip”, Kluwer Academic Publishers, Boston, pp. 53-47, 2002.

[12] Tei- Wei Kuo, Jan Wu, Hsin- Chia Hsih, “Real-Time Concurrency Control in a Multiprocessor Environment”, IEEE Transaction on parallel and distributed system, Vol 13, No. 6, June 2002.

[13] Chen, Tripathy and Blackmore, “A Resource Synchronization Protocol for Multiprocessor Real-Time Systems”, International conference on parallel processing, 1994.

[14] Chia-Mei Chen and Satish K. Tripathi, “Multiprocessor Priority Ceiling Based Protocols”, Thesis work, University of Maryland, April 7, 1994.

[15] R. Rajkumar. “Synchronization in Real-Time Systems: A Priority Inheritance Approach”. Kluwer Academic Publishers, 1991.

[16] Lui Sha, R. Rajkumar, “Priority Inheritance Protocols: An Approach to Real-Time Synchronization”, IEEE Transaction on Computers, Vol. 39, No. 9, September 1990.

**Authors’ Profiles**

**Mrs. Leena Das** is an Asst. Professor in School Of Computer Engineering at KIIT University, Odisha. In teaching she has been focusing on software engineering concepts and Object Oriented concepts. In research, her current interests include real time scheduling, fault tolerance in multiprocessor real time system and testing on real time system. She has received her MS degree in System Software from BITS, Pilani, India. Currently she is pursuing her PhD under KIIT University. She is a life member in ISTE.

**Miss Susmita Saha** has degrees BE in Computer Science Engineering from Burdwan University, W.B. She is currently pursuing M.Tech from the Department of Computer Science Engineering, KIIT University, Odisha.

Manuscript received 23rd Feb, 2014; revised 6th march, 2014; accepted 2nd april, 2014.