Memory Management: Challenges and Techniques for Traditional Memory Allocation Algorithms in Relation with Today's Real Time Needs

Muhammad Abdullah Awais
MS150200157, Virtual University of Pakistan
ms150200157@vu.edu.pk

***Abstract*– In recent era of computing, applications and operating systems cannot survive without efficient memory management, especially if an application has to be under severe load for undefined long time. Resources must be utilized efficiently to enhance performance. Real time systems require timely and proficient use of memory to perform efficiently otherwise the purpose of real time systems would be lost. It's the responsibility of operating system to provide the support for memory management through different ways supported as it acts as an interface between the primary resources such as hardware and applications running. Different memory allocation algorithms have been devised to organize memory efficiently in different timestamps according to the needs and scenario of usage yet there are issues and challenges of these allocators to provide full support for real time needs. Memory management in any operating system is governed by different aspects such as on hardware level, application level and especially the operating system level memory management which is our focus. Real time systems require memory on priority otherwise program may crash or may be unresponsive if demanded memory is not allocated with quick response. Beside the timing constraints, memory allocator algorithms must minimize the memory loss which comes in the form of fragmentation, the unusable memory in response to the memory allocation needs because memory is allocated in the form of blocks. Also the maintained locality of reference between memory blocks must be efficient for any memory allocation algorithm. Literature available provides extensive knowledge about memory allocation algorithms to satisfy the needs of real time applications. Our focus would be to analyse traditional dynamic memory management algorithms with respect to their functionality, response time and efficiency to find out the issues and challenges with these allocators to sum up the knowledge to know the limitations of these algorithm which might reduce the performance of real time systems. This research paper will give a comparative analysis of some well known memory management techniques to highlight issues for real time systems and innovative techniques suitable for these applications will be argued.**

***Keywords*– Memory Management, Dynamic Memory Management, Dynamic Memory Allocation, DMA, Real Time System, Operating System Memory Management, Fragmentation and Memory Blocks**

I. INTRODUCTION

Modern operating systems provide efficient memory management and still research is being conduct to improve the way the memory is allocated for applications because the main problem faces by memory allocation algorithm is to efficiently allocating the demanded memory blocks to the demanding applications with minimum response time along with minimum memory loss in the shape of traditional memory loss problem called the fragmentation of memory which keeping the reference to those blocks that has been allocated and to those blocks also which are free to be allocated for next demand by any application running on the operating system.

It's not enough to just provide the memory blocks needed by the application rather the efficiency of real time systems rely on the timely availability of these memory blocks with minimum fragmentation. For this purpose different kind of memory allocation designs are being utilized such as the static memory allocation and dynamic memory allocation as described in Fig. 1.

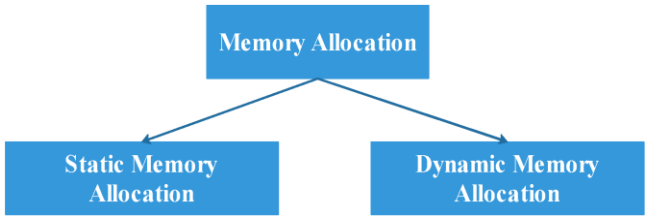

Fig. 1. Memory Allocation

Both these techniques are supported by real time systems and both of them differ the way the memory is distributed as in static memory allocation, memory is allocated at compile time and its known in advance what to allocate while in dynamic memory allocation scheme, the memory is allocation at run time and reference is maintained for allocated and unallocated memory blocks in the form of free and in use memory blocks. With the presence of these techniques, today's state of the art operating systems utilize dynamic memory allocation schemes through various different ways such as programming interface.

In the presence of different memory management techniques, goal of any memory allocation algorithm rest in providing real time support for memory allocation. Every memory allocation technique has its own pros and cons and it justify their performance for the purpose these techniques are

developed. Our intent is to figure out what these techniques can do and what is required by real time systems.

This research paper is divided in different sections where our intent is to analyze different traditional dynamic memory allocation algorithms to find out their response times and viability of these algorithms against real time applications. In section II of the paper, some related work and background knowledge will be presented. Section III will present research methodology followed by which it's possible to sum up this knowledge to comparatively analyze these techniques. In section IV different memory allocation algorithms will be presented along with their comparative analysis in next section and also a new technique suitable for real time applications will be discussed. In final section conclusion and suggestions with future work will be presented.

II. BACKGROUND KNOWLEDGE AND RELATED WORK

Extensive literature review revealed that researchers has indicated lot of limitations of traditional memory allocation techniques with justification and suggested improvements. Still research is being conduct because of the criticality of this topic. Real time systems have always been under research because of the constraints they impose such as quick response time required by real time systems, preemptive scheduling, and time based scheduling. These features of real time systems make them special and to serve them special allocators are devised to satisfy timely requests.

Dynamic memory management plays important role in memory management because of overhead associated with static memory management because whole required memory is allocated to running program at compile time and any block of that memory which is not used by application cannot be used by other application which is not efficient use of resources and further more dynamic memory allocation utilize heap memory data structure while stack is used in static memory allocation which makes DMA more efficient as compared to static memory allocation as discussed in [1].

In [2] a new variation of famous buddy system has been proposed called tertiary buddy which is an extension to binary buddy system with improved splitting and response time as compared to other buddy system variations. An overview of tertiary buddy will be presented in upcoming sections.

A lot of research has been conducted on improving dynamic memory allocators and the basics of segregated and sequential fit are always in research zone to be improved. Two level segregated fit algorithms is one of the improvements of segregated fit algorithm by [3]. While keeping in mind the requirements of real time systems, two levels segregated fit algorithm has been proposed. Even some improvements have also been done on two levels segregated fit algorithm to make it more suitable for real time systems by XiaHui and JinLin Wang.

Similar sort of work has already be done in [4] where author surveyed various techniques and algorithms in dynamic memory management and compiled result based on comparison but our work is different as I will include some new techniques and some more numerical analysis then in [5].

III. RESEARCH QUESTION

RQ: What are challenges and issues associated with traditional memory management techniques which hinders the performance in real time systems?

IV. RESEARCH METHODOLOGY

To answer the question on which my research is based, I performed extensive literature review according to the research guidance provided by B.Kitchenham [5]. According to the guidelines and research methodology I searched different research papers on the topic of memory management techniques. There is a bulk of data available online presenting different techniques for memory management in operating system. So in first search I found many research papers then I shortlisted some of them fulfilling my research topic. Many research papers are presenting comparative studies while in some papers, new techniques for memory management are proposed.

A. Searching Strategy

Initially I searched for memory management techniques in operating system to broaden and enhance my understanding about memory management so that essential concepts and ideas might not miss. To make sure I get relevant research papers with detail analysis of emerging memory management techniques , every possible search was conducted in IEEE explore digital library , Google scholar and third part research paper providing libraries such as Research Gate. To get relevant research knowledge I used keywords like memory management, memory allocation in operating system, real time operating system memory allocation, issues in memory allocation and techniques for dynamic memory allocation. By researching on different research publishing platforms, I got extensive data about operating system memory management techniques, allocators, algorithms and issues related with these techniques.

B. Selection

After studying basic of operating system memory management it was necessary to shortlist research papers on operating system memory management and issues related to traditional memory management techniques and reasons why these techniques are not best used for today's real time memory usage for applications and operating system which reduced number of research papers.

C. Study Methodology

Instead of pure comparative analysis of operating system memory management techniques, main focus was on understanding the operating system memory management techniques and to understand the situations in which any technique is applied. So to focus on the result an overview and essential detail of some new and already used techniques is presented in this paper and key issues related to these algorithms are summed up to conclude the complexities involved with these techniques and requirements for real time

applications to answer the research question. Fig 2 shows basic model followed for the research paper.

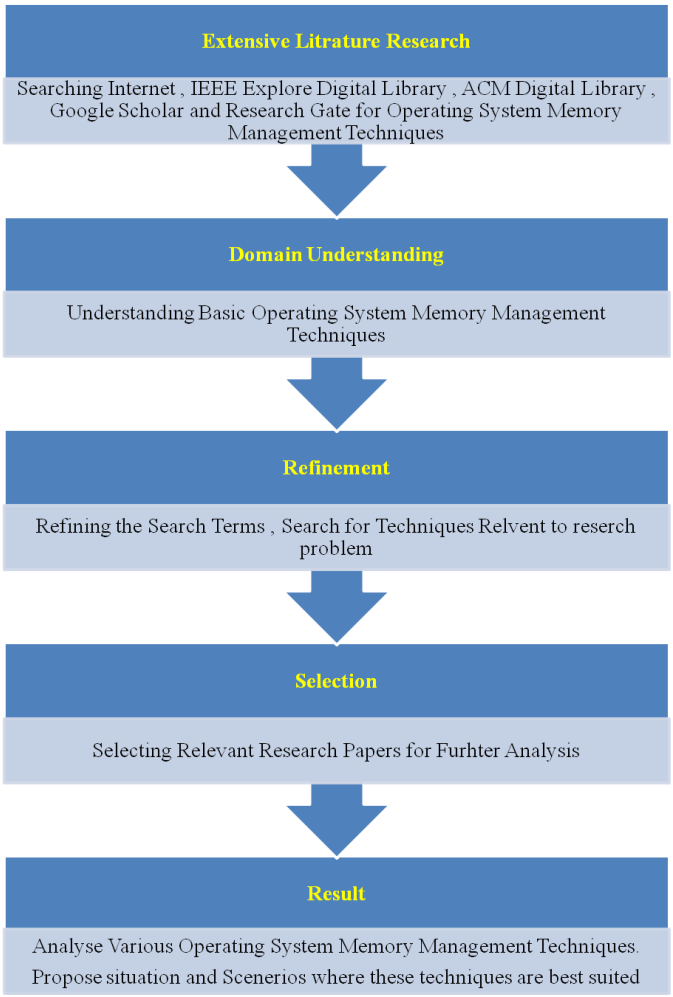

Fig. 2. Basic Model for Research

V. DYNAMIC MEMORY MANAGEMENT ALGORITHMS

Due to the significance of dynamic memory management in operating system, most of the traditional and new memory allocation algorithms utilize dynamic memory allocation scheme to allocate memory from heap at run time as explained in [6]. Here in this section we will provide an overview of traditional algorithms under dynamic memory allocation because it's the scheme which is utilized by state of the art real time systems and has excellent operating system support. New algorithms are devised based on the limitations of previous algorithms and with improvements so we will fist discuss traditional algorithms then we will have a look on new algorithms devised for real time systems.

A. Sequential Fit

As the name suggest, this algorithm utilize the free blocks of memory in linear order in the form of a list called free list. And memory blocks are allocated from this free list using pointer in different ways according to the situation in hand. There are four different strategies used by sequential fit algorithm as discussed below and difference is shown by Fig. 3.

1) *First fit:* First fit is the simplest strategy followed by sequential fit as the first available memory block which is greater or equal to demanded memory is served irrelevant of the consequences.

2) *Next Fit*: Next fit is similar to first fit but it start searching the list from the position where last search stopped and it serve the next available memory block.

3) *Best Fit:* As name suggest, best fit will allocate that block which is best in terms of demanding size.

4) *Worst Fit*: It's opposite to best fit as it will always return the largest memory block available.

In Fig. 3 sequential fit algorithms is shown in action. Red block indicate the memory blocks already used and are not available to be used while available memory blocks are ladled with the capacity. Current pointer position is shown after first 1k memory. Now we will show the execution of this algorithm if 2k memory is demanded by application.

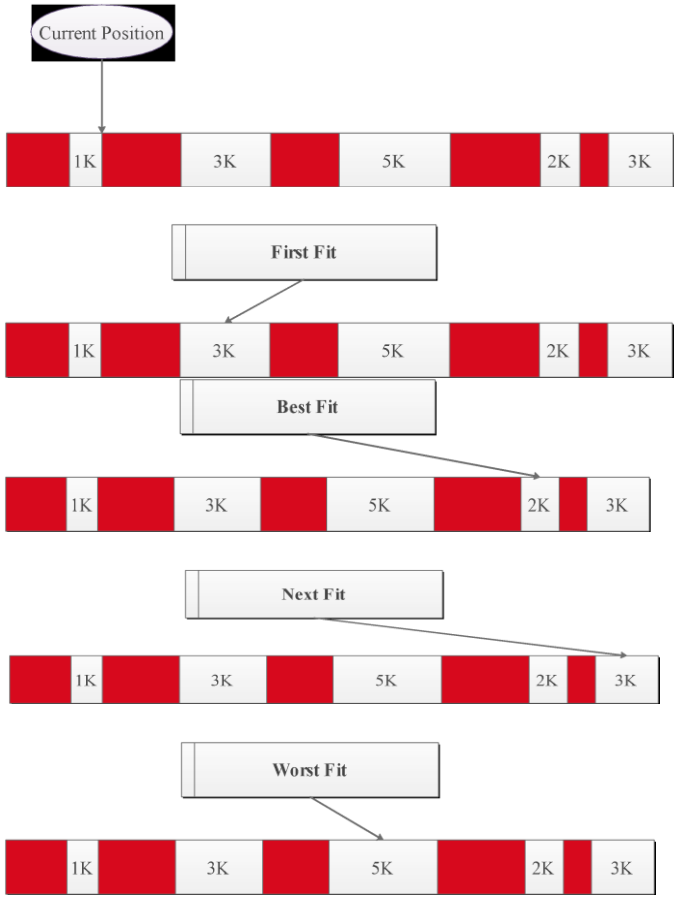

Fig. 3. Sequential memory blocks

According to the scenario current pointer position is after the 1k memory location as indicated in Fig. 3. If first fit is used then the very first memory block from the current pointer which can satisfy the demand is served. While in best

fit, that memory block will be served which minimize the memory wastage while worst fit will always return the largest memory block.

B. Segregated Fit

Segregated fit algorithm employ array of free blocks to allocate memory and this methodology is also incorporated by many advanced memory allocators. Main theme of segregated free list algorithm is to use size in power of two [7]. And divide memory blocks into classes holding different size blocks. By this way whenever a request of particular size is received, segregated algorithm round the size of that request up to the best available class of particular memory blocks and then memory block from matching class size is allocated. Simple logic behind this technique is shown in Fig. 4. Like sequential fit algorithm, segregated fit algorithm also employs certain strategies as discussed below.

1) *Strict Size classes:* Basic idea behind this kind of strategy is to maintain a list of different classes holding memory blocks of similar sizes. That's way each class of particular size will hold memory blocks of same size in list.

2) *Exact List*. This strategy involves in marinating large number of free lists of all possible memory block sizes and it's best used if there are small size classes containing free lists of huge number.

3) *Classes with Range:* In this type of segregated free list, free list may contain different size blocks.

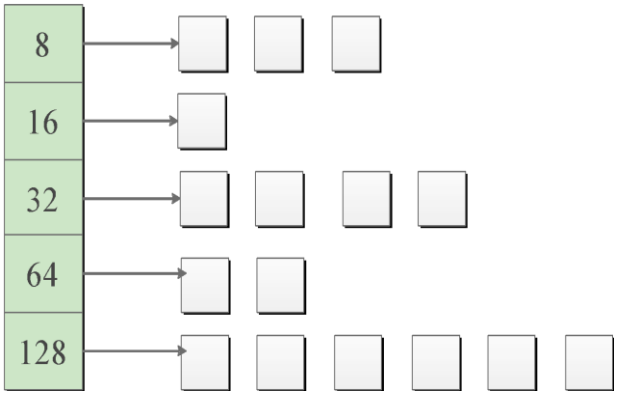

Fig. 4. Segregated free list

C. Buddy System

Buddy system is innovative way of memory allocation based on the idea behind segregated free list methodology where size of classes is used with rounding. These way free lists are separated according to sizes. In simple words it divides the memory area into allowable block size and partition the area until minimum block size is achieved. In Fig. 5 basic operation of buddy system is shown where a 3k memory needs to be allocated and it partition the available memory and allocate this memory block.

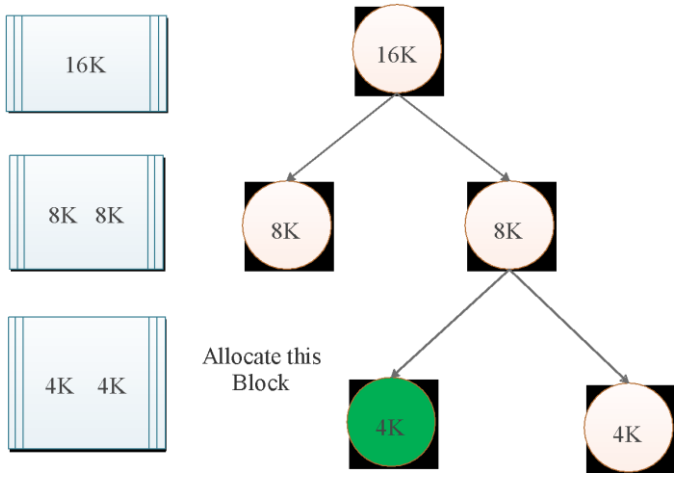

Fig. 5. Basic Buddy system

1) *Binary Buddy:* In binary buddy variation, all block sizes preserve the property of power of 2 and splitting of memory in 2 equal halves is observed in binary buddy.

2) *Weighted Buddy*: Like binary buddy version, weighted buddy also exhibit power of 2 scenarios but splitting can take place in 2 equal halves or 2 unequal halves because series can be power of two and 3 times the power of two as shown in Fig. 6.

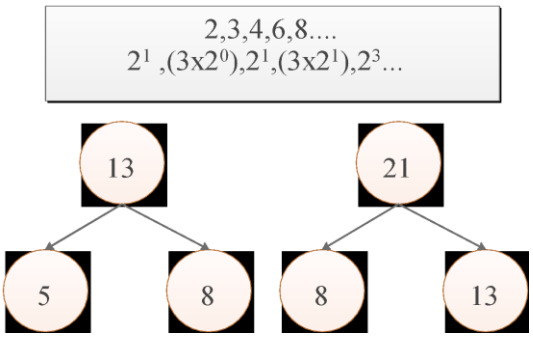

Fig. 6. Weighted Buddy system

3) *Fibonacci Buddy:* According to the name, Fibonacci buddy follow the ancient Fibonacci sequence and size classes are based on Fibonacci sequence [8] as in Fig. 7.

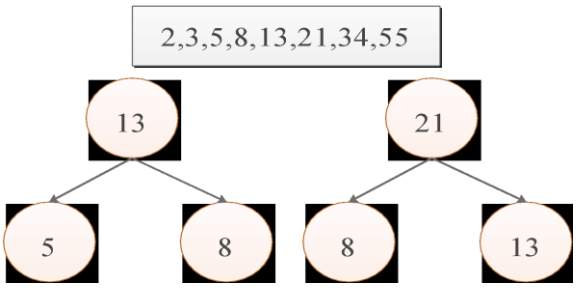

Fig. 7. Fibonacci Buddy system

4) *Double Buddy:* Just like binary buddy and weighted buddy in this variation there are 2 classes, one following the rule of power of two while in other list there is power of 2 and offset value is used.

5) *Tertiary Buddy:* It's an extension to binary buddy. In tertiary buddy block sizes are power of 2 and 3 x 2^{x-3}. By this variation its far more better than binary buddy as detailed analysis provide by [3]

VI. COMPARISON OF BUDDY SYSTEM VARIATIONS

In previous section we have discussed different versions of buddy system. Binary buddy is very simple and due to the equal size partition make it easy to compute pointer which makes this buddy allocator a real time allocator. Despite of this advantage internal fragmentation is on higher side as compared to others. On the other hand Fibonacci buddy has lower internal fragmentation than binary buddy while weighted buddy with different classes has lower internal fragmentation than all other buddy system variations. In Fibonacci buddy block splitting only take place if sizes are in numbers. Results of fragmentation are publicized in table 1 below.

TABLE 1
Comparison of Different Buddy system schemes

SN	Buddy System Variations		
	Binary	**Internal Fragmentation**	
1	Binary	Higher than others	
2	Double Buddy	Lower than Binary Buddy	
3	Fibonacci Buddy	Lower than Double Buddy	
4	Weighted Buddy	Lowest	
5	Tertiary Buddy	Lowest than all of Buddy Variations	

D. Indexed Fit

In Indexed fit memory allocator an index of free and reserved memory blocks is maintained using different types of data structures. Indexing is employed in any other technique in several ways because it's the most basic mechanism for traversing or searching an array or list. As far as response time is concerned it is somewhat faster than traditional sequential fit algorithm. Fig. 8 shows basic indexing layout.

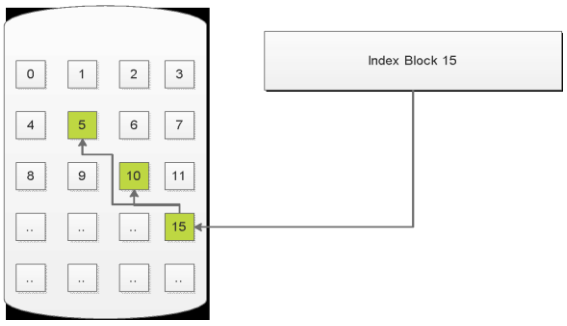

Fig. 8. Indexed fit

E. Bitmapped Fit

Bitmapped fit is an improved variation of indexed fit and it keeps references to the used and free portion of array by using bits. Due to searching time which is quite high, bitmapped is not used as much as other allocators are used. Yet research is being conducted on improved versions of bitmapped allocation algorithm because in new operating systems and applications there are situation where bitmapped fit can be efficient to use.

F. Half Fit

Half fit is much older technique which used bitmaps to keep reference to unfilled lists while using instructions of bitmap search technique to get those bits which are set in bitmaps. Although it's known that bitmap is little bit slower but while combining and improving, it gave good results. Main theme behind half fit is to use segregated list of single level which is used to link variable size free blocks. Fig. 9 shows implementation details of half fit in action.

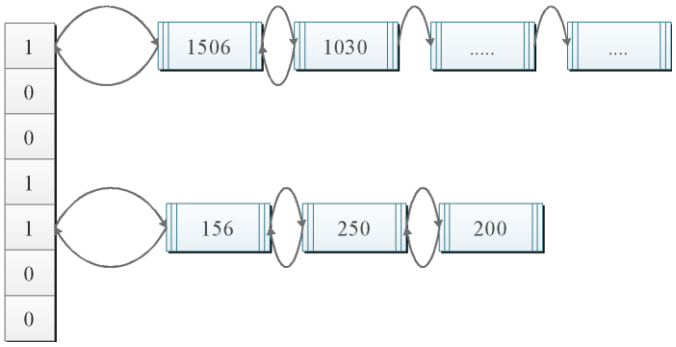

Fig. 9. Half fit blocks 156,250 and 200

G. Hoard

Hoard is designed especially for multiprocessor systems and its performance is quite remarkable among other discussed algorithms. Trick logic behind hoard is to use operating system virtual memory as superblocks and these superblocks are used to server blocks of memory of one class. To reduce external fragmentation it re cycle its superblocks which are not in use [9].

VII. TWO LEVEL SEGREGATED FIT

It's an important algorithm in modern dynamic memory allocation. It stems its root from segregated fit and half fit as described earlier. It's different from traditional hoard algorithm because it uses segregated lists in 2 levels as its name suggest. These 3 levels of segregated free lists are used to carry free blocks of memory of same class which reduce internal fragmentation. In first level there are free blocks of memory following power of 2 sequences while 2^{nd} list uses user's configured variables to divide free block classes of first list. Thus help to offer bounded response time. While allocating and de allocating it uses 3 different equations as described in [11] with essential implementation detail while Fig. 10 shows basic graphical view of two level segregated fit

algorithm. Performance and working analysis is presented in section VIII.

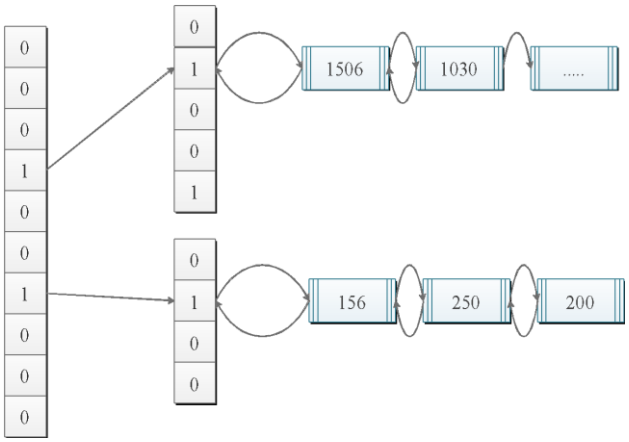

Fig. 10. Two Level Segregated fit

VIII. COMPARATIVE ANALYSIS

In previous sections we have discussed some traditional memory allocation algorithms. Here in this section a comparative analysis is presented with respect to allocation and de allocation time of different algorithms. Then a comparative analysis with respect to fragmentation and response time will be presented.

Algorithms	Allocation time	DeAllocation Time
TLSF	O(1)	O(n)
Half Fit	O(1)	O(1)
Hoard	O(n)	O(1)
Tertiry Buddy	O(1)	O(n*)
Bitmapped	O(n)	O(1)
Buddy systems	O(log2n)	O(k)
Segregated Fit	O(1)	O(1)
Sequential Fit	O(n)	O(1)

Fig. 11. Worst Case Time Complexity

First of all the sequential fit algorithm is slow because it has to traverse the list if following the best fit strategy to find the optimal memory block which minimize the fragmentation and this algorithm is implemented by famous doubly linked list. Best part of best fit algorithm is that it minimizes the fragmentation as if found memory block is optimal and yet larger than requirement then splitting of block take place to use the required space and remaining is freed immediately. Same way first fit and next fit works by splitting of blocks but following their underlying strategy as described in previous sections. Overall allocation and de allocation time of sequential fit is compared with other algorithms in Fig. 11 while major drawback of this algorithm is the amount of fragmentation it cause and the response time as in Fig. 12.

Segregated free list is one of those algorithms which have been used to devise more advanced and optimal algorithms such as hoard and two level segregated fit. In its pure form its performance is not as good as if it is used in conjunction with other algorithms because on its own it causes large fragmentation with maximum memory trace. On the other hand performance of indexed fit is somewhat similar with bitmapped and segregated fit algorithm.

Among all these memory allocators, performance of two level segregated fit is better because its worst case time is less than other's while it also minimize the fragmentation with fast response time which makes it suitable for real time application.

Algorithm	Fragmentation	Response	Memory Footprint
Buddy System	Large	Fast	Max
Sequential Fit	Large	Slow	Max
Segregated Fit	Large	Fast	Max
Index Fit	Large	Fast	Max
Bitmapped Fit	Large	Fast	Max
TLSF	Smaller	Fastest	Min
Hoard	Small	Faster	Min
Tertiary Buddy	Small	Fast	Min

Fig. 12.

IX. CONCLUSIONS AND SUGGESTIONS

In this research paper different memory allocation techniques have been discussed along with their comparative analysis with respect to internal fragmentation they cause, response time, allocation time, de allocation time and memory footprint they use. Every technique discussed belonging to dynamic memory management has pros and cons and can be best utilized in particular situation. Most of the algorithms are improved versions of previously discussed schemes such as sequential and segregated fit and TLSF. Analysis shows that TLSF among mentioned technique is best to use for real time systems because TLSF cause very low internal fragmentation, its response time is very good which is the primary demand of real time system where time is most important factor. Also TLSF allocation and de allocation time is small constant time that makes it much faster than other traditional techniques.

With comparative analysis it's found that the larger fragmentation, slow response time, larger allocation and de allocation time with implementation constraints, it makes traditional dynamic memory allocators like segregated fit, indexed fit, bitmapped fit and simple buddy system in feasible and in efficient for real time system because real time systems always pose timing and bounded rationality constraints on

operating system memory management allocators. So Hoard, tertiary buddy system and two level segregated fit are suitable for real time applications with faster response time, minimum amount of fragmented memory respectively.

REFERENCES

[1] Nilesh Vishwasrao and Prabhudev Irabashetti, "Dynamic Memory Allocation: Role in Memory Management", International Journal of Current Engineering and Technology, Vol. 4, No. 2, April 2014.

[2] Divakar Yadav and Ashok Sharma, "Tertiary Buddy System for Efficient Dynamic Memory Allocation", Conference: Proceeding SEPADS'10 Proceedings of the 9th WSEAS international conference on Software engineering, parallel and distributed systems, At Cambridge.

[3] Masmano, I.Ripoll, A. Crespo, and J. Real," TLSF: a new dynamic memory allocator for real-time systems", Real-Time Systems, 2004. ECRTS 2004. Proceedings. 16th Euromicro Conference

[4] Dipti Diwase, Shraddha Shah, Tushar Diwase and, Priya Rathod, "urvey Report on Memory Allocation Strategies for Real Time Operating System in Context with Embedded Devices", International Journal of Engineering Research and Applications (IJERA), Vol. 2, Issue 3, May-Jun 2012, pp.1151-1156.

[5] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Version, Vol. 2, 2007, pp. 2007–01.

[6] Puaut, "Real-Time Performance of Dynamic Memory Allocation Algorithms,"14th Euromicro Conference on Real-Time Systems (ECRTS'02), June 2002.

[7] Mohamed A. Shalan, "Dynamic Memory Management for Embedded Real-Time Multiprocessor System On a Chip", A Thesis in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy from School of Electrical and Computer Engineering ,Georgia Institute of Technology, November 2003.

[8] B. Cranston and R. Thomas, "A Simplified recombination Scheme for the Fibonacci Buddy System," CACM, June 1975, 331-332.

[9] Valtteri Heikkilä, "A Study on Dynamic Memory Allocation Mechanisms for Small Block Sizes in Real-Time Embedded Systems", University of Oulu Department of Information Processing Science, conference 17, December 2012.

[10] Takeshi Ogasawara, "An Algorithm with Constant Execution Time for Dynamic Storage Allocation", Advanced Compiler Group Tokyo Research Laboratory, IBM Japan, Ltd. 1623-14, Shimo Tsuruma, Yamato-Shi, Kanagawa 242, Japan.

[11] Seyeon Kim, "Node-oriented dynamic memory management for real-time systems on ccNUMA architecture systems", University of York Department of Computer Science, conference paper April 2014.