# Tailed Block Group System: An Efficient Dynamic Memory Management Scheme for Resource-constrained Real-time Operating Systems

Sung Ho Park', Yu Jin Park[2], Soon Ju Kang[2]

[1] IDIS, IDIS Tower 688-1, Sampyeong-dong, Bundang-gu, Seongnam, Gyeonggi-do, Korea, slblue@ee.knu.ac.kr

[2] School of Electronic/Electrical Engineering and Computer Science, Kyungpook National University, 1370 Sankyuk-dong, Buk-gu, Daegu, Korea, {esssusige, sjkang}@ee.knu.ac.kr

Abstract. In this paper, we propose a new dynamic memory management scheme named tailed block group system (TBGS). It provides bounded execution time. The complexities of its allocation and release algorithm are *O(log N)*, where *N* is the total dynamic memory size. In addition, it has high memory efficiency. Its memory efficiency is close to that of the best fit, one of the most memory-efficient dynamic memory management schemes. Therefore, TBGS can be a better dynamic memory management solution for resource-constrained real-time operating systems than existing dynamic memory management schemes can.

Keywords: Dynamic Memory Management, Real-Time Operating System.

## 1 Introduction

The need for a high performance dynamic memory management scheme is increasing due to the advances in software engineering, such as graphical user interface, and object oriented languages [1-2]. This trend occurs not only in the general purpose systems but also in resource-constrained real-time operating systems [2-3]. However, existing dynamic memory management schemes do not provide sufficient performance [4-6]. In particular, it is hard to meet the requirements of the resource-constrained real-time operating systems in existing memory management schemes [7]. A resource-constrained real-time operating system requires bounded execution time, and generally has limited memory resource. However, the existing schemes cannot provide bounded execution time and high memory efficiency simultaneously [5, 6, 8-11]. Schemes that provide bounded execution time have low memory efficiency, and the memory-efficient schemes have unbounded execution time.

In this paper, we propose a new dynamic memory management scheme named tailed block group system (TBGS). It provides bounded execution time. The complexities of its allocation and release algorithm are *O(log N)*, where *N* is the total dynamic memory size. In addition, it has high memory efficiency, because fragmentation caused by it is very small. The rest of this paper is organized as follows: Section-$_{1}$**q$_{r}$! ti$^{-}$2.S a z ;tit slog F.** introduces related work. Section 0 explains TBGS. Section 2-$_{1}$7-! o$^{1}$-2E

**,sy z ;4-* $Tg_A$Li El.** evaluates TBGS and typical existing dynamic memory management schemes. Finally, section 0 concludes this paper.

## 2 Related work

### 2.1 Existing dynamic memory **management schemes**

Existing dynamic memory management schemes are classified into sequential fits, indexed fits, segregated fits, buddy systems, and hybrid schemes [5, 6]. Best fit and first fit are typical sequential fit schemes [5, 6]. They have high memory efficiency, but their execution time is unbounded. Indexed fits [5, 6, 12] are varieties of sequential fits. The ordered binary tree best fit and the Cartesian tree best fit are typical schemes of this class. Their average execution time is shorter than that of sequential fits, but their execution time is also unbounded. Fast fit is the typical scheme of segregated fits [5, 6]. It has short and bounded execution time, but its memory efficiency is low. Buddy systems [5, 6, 11] are varieties of segregated fits. They also have low memory efficiency, although they do better than fast fit does. Some popular operating systems, such as Microsoft Windows and Linux, use hybrid schemes, because it is hard to provide sufficient performance with one of the existing dynamic memory management schemes. Their default dynamic memory allocators use a hybrid scheme of segregated fits and sequential fits, with optimization for the common dynamic memory usage patterns. They have good average performance, but their execution time is unbounded.

### 2.5 Fragmentation

Fragmentation [14] is generally used as the metric for memory efficiency of dynamic memory management schemes, because it is the dominant cause of the wasted memory in dynamic memory management. In this paper, we also use it as the metric for memory efficiency. There are two types of the fragmentation. The first is internal fragmentation. In some dynamic memory management schemes, the allocated block size can be larger than the request size, because the allocable sizes are limited. The memory wasted due to the over-allocation is internal fragmentation. The second is external fragmentation. Although the total available memory is larger than a request size, the request can fail, because there is no consecutive available memory enough for the request. The memory wasted due to this phenomenon is external fragmentation.

## 3 The proposed scheme: Tailed Block Group System (TBGS)

In this section, we will explain the tailed block group system (TBGS), a new efficient dynamic memory management scheme proposed in this paper. The key ideas of TBGS are the group system (GS), and the tailed block (TB). The GS is our initial design. It has bounded execution time, and enhanced memory efficiency compared to existing schemes with bounded execution time. However, there is a big deviation in its memory efficiency

according to the memory usage pattern. Thus, we redesigned GS to the TBGS. TBGS solves the problem with the TB. Section 0 details GS. Section Q**om! tE fj § § (.1 LI EF.** explains the TB concept and introduces TBGS.

### 3.1 Group System (GS)

GS is a variety of segregated fits. It supports limited block splitting and coalescing, like the buddy systems [11]. The boundary tag[13] is used for block coalescing. It has a limited set of allocable sizes. The size of a block made by splitting or coalescing is always included in the set. Thus, it can manage available blocks with a fixed number of lists, although it holds available blocks in segregated lists, according to the block size and supports block splitting and coalescing. However, its block splitting and coalescing are more flexible than those of buddy systems are. Buddy systems split a block into two blocks by a fixed ratio, whereas GS splits a block into various numbers of blocks by various ratios. The number of allocable sizes of GS exceeds that of buddy systems due to this flexibility. However, the allocable sizes are not evenly distributed. Thus, there is a big deviation in its memory efficiency, although its average memory efficiency exceeds that of buddy systems.

Let *M* be the maximum number of members of GS. Then, a parent block can be split up into *M* child blocks, and the size of the child blocks can be a multiple of *(parent _block_size÷ M)* Child blocks that split from the same parent block belong to the same group, and blocks belonging to the same group are coalesced when they are released. A child block of size *(parent_block_size÷ M)* becomes a parent block, and the block splits into child blocks, again.-$_1$9-! 11101-1 shows an example of memory allocation with the GS whose *M* is 4. The complexity of GS is *O(log N),* where *N* is the total dynamic memory size.

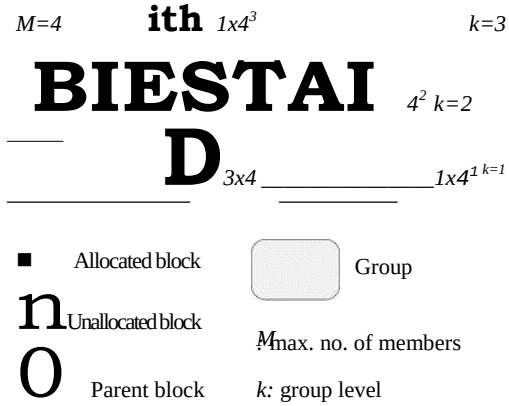



**Fig. 1 Example of memory allocation with the group system (GS). 3.2 Tailed Block Group System (TBGS)**

TBGS is an enhanced group system solving the big deviation in memory efficiency. TBGS uses the TB concept to solve the problem. In addition, the TB concept improves the memory efficiency itself. The concept is adding small size tails to the left and the right

of a block. This increases the number of allocable sizes and spreads them evenly. The definition of the body part is the same as that of the block of GS. The size of the tail part can be a multiple of $(parent\ block\ size \div M^2)$ $2.\%!$ $\mathbf{t_{i \cdot 3s}}$

**$p_u$i,e, Li ci-.2** shows an example of memory allocation with TBGS whose *M* is 4. The complexities of TBGS is also *O(log N)*, where *Nis* the total dynamic memory size.

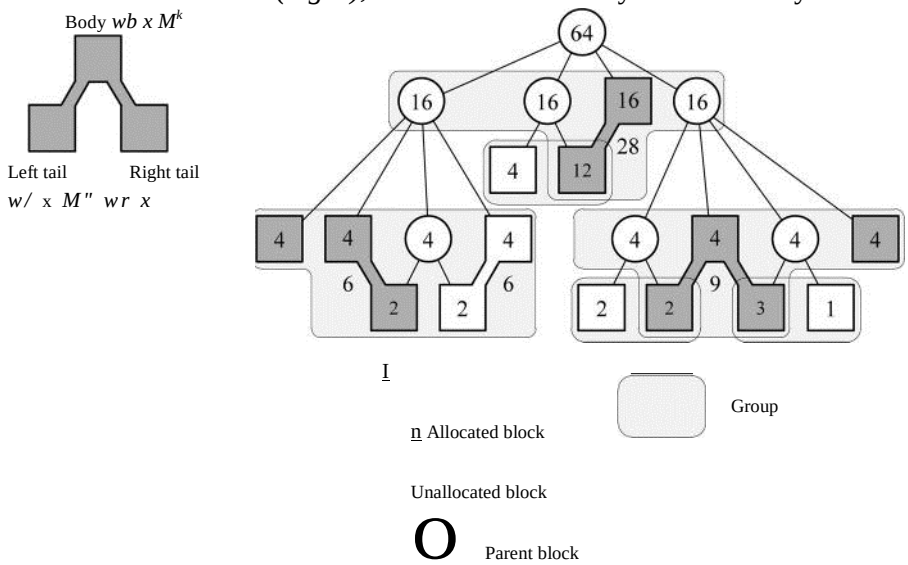



**Fig. 2 Example of memory allocation with the tailed block group system**

**(TBGS). 5 Performance evaluations**

In this section, we evaluate GS, TBGS, and typical existing dynamic memory management schemes. The evaluation was based on Peterson and Norman's work [11] that tested various buddy systems with the Monte Carlo method. We defined the internal, external, and total fragmentation ratio as equations *(1), (2),* and *(3).*

$$internal\ fragmentation\ ratio = \frac{total\ allocated\ memory - total\ requested\ memory}{total\ allocated\ memory} \quad (1)$$

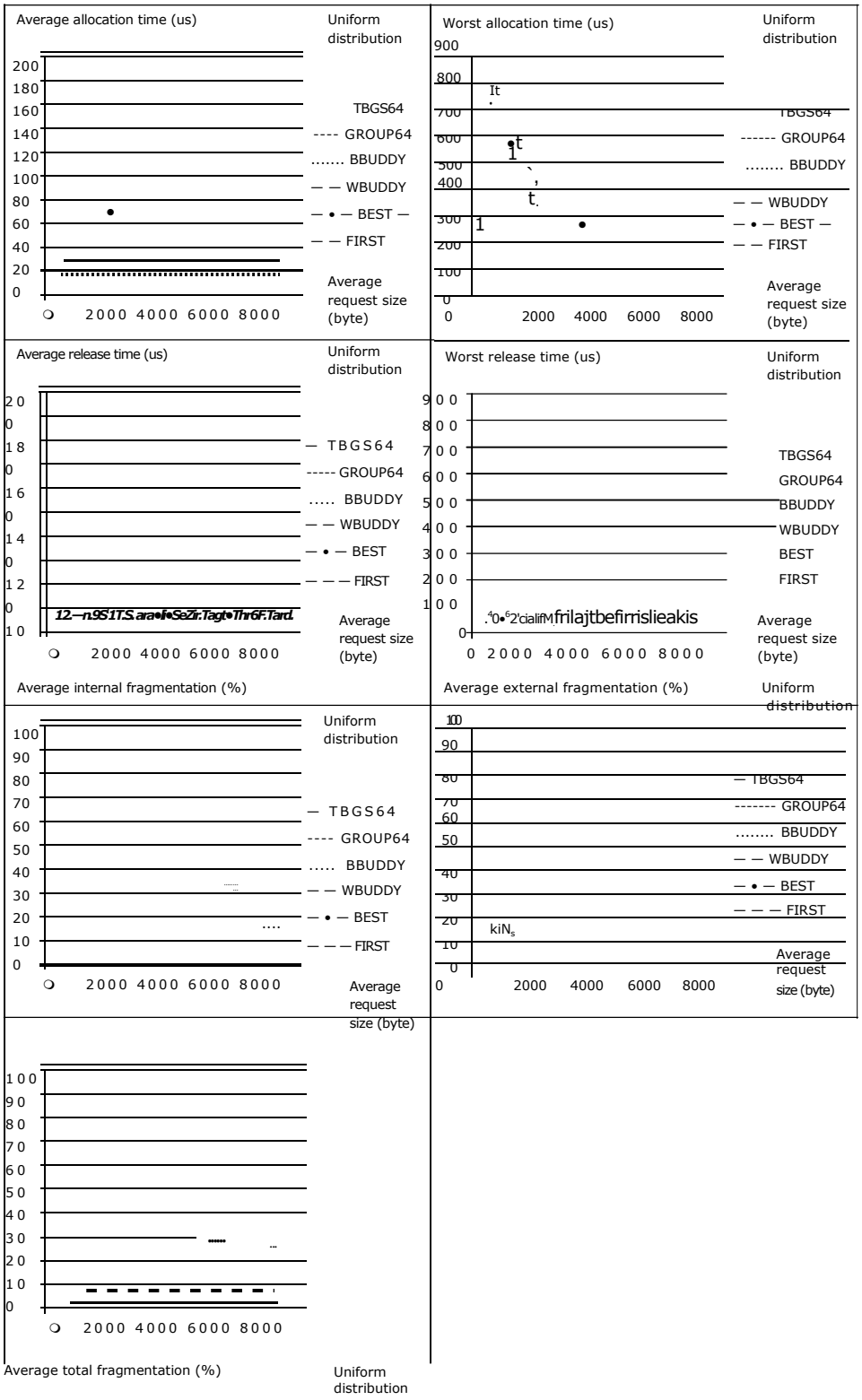
$$external\ fragmentation\ ratio = \frac{total\ dynamic\ memory - total\ allocated\ memory}{total\ dynamic\ memory} \quad (2)$$

$$total\ fragmentation\ ratio = (1 - external) \times internal + external \quad (3)$$

The size and lifetime distributions used in this evaluation were uniform, exponential, and normal distribution. The evaluation was performed on the 200 MHz ARM9 machine running Ubinos [15]. Ubinos is a multi-threaded operating system developed by us for resource-limited real-time embedded systems. The 8 Mbytes independent memory region was used as the dynamic memory for the evaluation. MMU and cache were disabled.

24! § **4** C. show the evaluation results for the uniform size distribution and the uniform lifetime distribution. These results show the execution time of TBGS is short and bounded, like those of buddy systems. Also, they show the best

fit has the best memory efficiency, and the memory efficiency of TBGS is close to the best memory efficiency. Evaluations using other size distributions and lifetime distributions also show similar results.




TBGS64

------ GROUP64

..... BBUDDY

— — WBUDDY — • — BEST

— — — FIRST

Average request size (byte)

Fig. 3 Evaluation results for the uniform size distribution and the uniform lifetime distribution.

## 6 Conclusion

In this paper, we proposed a new efficient dynamic memory management scheme for resource-constrained real-time operating systems. The new scheme, termed TBGS, has short and bounded execution time and high memory efficiency, simultaneously. Our evaluation results confirm that the execution time of TBGS is short and bounded, and the fragmentation caused by TBGS is very small. The total fragmentation ratio of TBGS is close to that of the best fit, one of the most memory efficient dynamic memory management schemes. Therefore, TBGS can meet the growing demand for efficient dynamic memory management in resource-constrained real-time operating systems better than the existing dynamic memory management schemes can.

**Acknowledgement.** This work was supported by the IT **R&D** program of MKE/KEIT. [10041145, Self-Organized Software-platform(SOS) for welfare devices

## References

1. J. M. Chang and C. H. Daugherty: An Efficient Data Structure for Dynamic Memory Management, Journal of Systems and Software, vol. 54, pp. 219-226 (2000)
2. Y. Hasan and J. M. Chang: A Tunable Hybrid Memory Allocator, Journal of Systems and Software, vol. 79, pp. 1051-1063 (2006)
3. S. M. Donahue, et al.: Storage Allocation for Real-Time, Embedded Systems, in Proceedings of 1st International Workshop on Embedded Software, pp. 131-147 (2001)
4. E. D. Berger, et al.: Composing High-performance Memory Allocators, ACM SIGPLAN Notices, vol. 36, pp. 114-124 (2001)
5. M. S. Johnstone and P. R. Wilson: The Memory Fragmentation Problem: Solved?, ACM SIGPLAN Notices, vol. 34, pp. 26-36 (1999)
6. P. R. Wilson, et al:, Dynamic Storage Allocation: A Survey and Critical Review, in Proceedings of the International Workshop on Memory Management, London, UK, pp. 1-116 (1995)
7. I. Puaut: Real-time Performance of Dynamic Memory Allocation Algorithms, in Proceedings of the 14th Euromicro Conference on Real-Time Systems, Washington, DC, USA, pp. 41-49 (2002)
8. N. R. Nielsen: Dynamic Memory Allocation in Computer Simulation, Communications of the ACM, vol. 20, pp. 864-873 (1977)
9. C. Bays: A Comparison of Next-fit, First-fit, and Best-fit, Communications of the ACM, vol. 20, pp. 191-192 (1977)
10. J. E. Shore: On the External Storage Fragmentation Produced by First-fit and Best-fit Allocation Strategies, Communications of the ACM, vol. 18, pp. 433-440 (1975)
11. J. L. Peterson and T. A. Norman: Buddy Systems, Communications of the ACM, vol. 20, pp. 421-431 (1977)
12. C. J. Stephenson: New Methods for Dynamic Storage Allocation (Fast Fits), in Proceedings of the 9th ACM Symposium on Operating Systems Principles, Bretton Woods, New Hampshire, United States, pp. 30-32 (1983)
13. D. E. Knuth: The Art of Computer Programming, Volume 1 - Fundamental Algorithms, 3rd ed.: Addison-Wesley Longman Publishing Co., Inc. (1997)
14. B. Randell: A Note on Storage Fragmentation and Program Segmentation, Communications of the ACM, vol. 12, pp. 365-372 (1969)
15. Ubinos - A Multi-threaded Operating System for Resource-limited Real-time Embedded Systems, http://www.ubinos.org